UNIVERSITAT DE BARCELONA

Trabajo de Fin de Grado

GRADO EN INGENIERIA INFORMÁTICA

Facultad de Matemáticas e Informática
Universitat de Barcelona

MONITORIZACIÓN, DETECCIÓN Y BLOQUEO DE PROCESOS DE CIFRADO MALICIOSO

Víctor Campos Martín

Tutor: Francesc Dantí
Realizado en: Departamento de Matemáticas e Informática. UB

Agradecimientos

Quiero agradecer a mis padres el apoyo y la ayuda que siempre me han dado para poder llegar hoy aquí.
A mi tutor, Francesc Dantí, todo el apoyo que me ha dado para llevar a cabo este proyecto, mostrando en todo momento la pasión, el interés y exigencia necesarios. Ha hecho que pueda estar enormemente orgulloso del trabajo realizado.
A Silvia quiero agradecerle las horas que ha estado escuchándome hablar sobre el tema, mostrando siempre su más sincero apoyo, por muy extraños que parecieran los conceptos que le explicaba.
Y por último agradecer a esos profesores que han priorizado ser apasionados antes que docentes. Los que han sabido ir más allá en sus clases y mostrar a los alumnos que la pasión y la curiosidad son imprescindibles para seguir el ritmo a la tecnología.

Sumario

Abstract

This project wants to give a solution to Ransomware, a problem that in 2016 is affecting the biggest amount of users in malware's world.

Ransomware is a kind of malware characterized by asking a ransom payment after infecting a device. Firstly they just block the device showing a full screen message until receiving the payment but, in a while, they started using file encryption.
Once the files have been encrypted, it is virtually impossible to decipher them without the decryption key. That leaves only the possibility of ransom to recover lost files.

During the investigation about ransomware, we found that the vast majority of them used fixed extensions and patterns to rename encrypted files. Somehow, we could use this feature to identify the encryption process in its initial state and kill it.

RaMON is a reactive tool that doesn't require installation and designed to consume very little resources. These characteristics make possible to work together with an antivirus as a light and transparent application.
We must remember that RaMON has been designed to fight against a very specific type of malware. For this reason, it should be viewed as an additional security layer and in no way a replacing for an antivirus.

RaMON has a *blacklist* with extensions we consider as IOC (Indicator of Compromise). When one of this extensions is detected, a malicious encryption process is taking place.
From them, the functionality of the tool follows these steps:

- Monitoring File System for detecting creation/rename of new executable files (.exe)
- Monitoring creation/rename files with dangerous extensions.
- Matching the “Last created EXE’s” list with current process list, in order to find encryption process.
- Once found, matches the “Last created EXE’s” list with current process list, in order to find encryption process. After that, it sends a kill signal to it, his sons and threads.
- In parallel, disables network interfaces to avoid expansion of the infection.
- Sends a shutdown informing the user about the infection. We make this in order to avoid to keep modifying the system, just in case of an eventual forensic analysis.

As a last line of defense tool, its performance will only take place if the ransomware has bypassed all other security layers (UAC, execution prevention, antivirus, firewall, etc.)

We should note that the computer world in general, and malware in particular, improves at high speed and what is effective today, probably tomorrow will not. The same sources of information serve the blackhat and the whitehat hackers, fueling the fast evolution in the world of security.

Most of time we are thinking about improving security applications but sometimes we forget to work hardly in user education, that is always the weakest link in the infection chain.

1. Introducción y antecedentes

El ransomware es un tipo de malware capaz de cifrar todos los archivos del equipo y de otros equipos a los que se tenga acceso desde él, para posteriormente solicitar un rescate por ellos.

Este tipo de malware se ha posicionado en los últimos años como la amenaza más activa debido a su elevado ratio de efectividad y a los beneficios que genera.

El malware clásico siempre se ha caracterizado por su comportamiento sigiloso. El pasar desapercibido era uno de los aspectos más importantes de la infección, puesto que esto les permitía mantener el acceso en el ordenador de la víctima durante más tiempo y poder con ello conseguir más información, documentos o claves. Por el contrario, el ransomware aparece de forma abrumadora en el equipo, bloqueando en ocasiones su uso y mostrando pantallas amenazadoras e intimidatorias al usuario. Para este fin, también se usan imágenes intimidatorias (haciéndose pasar por la policía), audios que anuncian al usuario que toda su información ha sido cifrada, imágenes de películas de miedo o contenido sexual, imágenes captadas por la webcam o imágenes de calaveras. Este nuevo comportamiento tiene como finalidad asustar e incomodar al usuario que, junto con la necesidad de recuperar su información y el hecho de que el precio del rescate sea en la mayoría de casos asequible (el precio medio pagado por los rescates se sitúa en 500€ [1]), hacen que se acabe realizando el pago del rescate en gran parte de las infecciones.

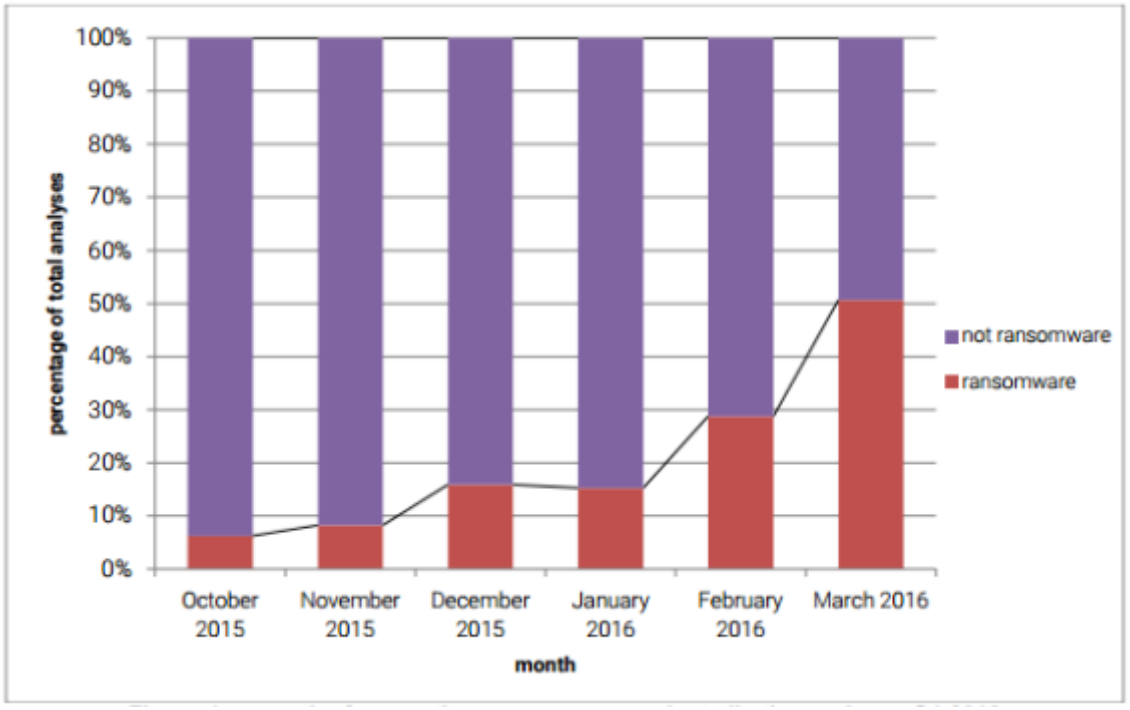


Figura 1. Crecimiento del Ransomware de cifrado en relación al resto de malware Q1 2016 [2]

En la Figura 1 podemos ver que en el total de infecciones por malware, el ransomware cada vez representa una parte más grande, debido a que se ha establecido como una de las infecciones con mayor rendimiento económico para los delincuentes.

Debido a la gran rentabilidad, los atacantes prefieren usar este tipo de infecciones.

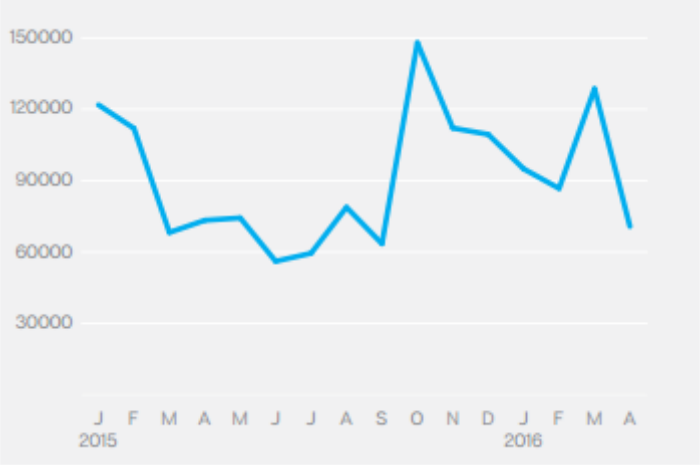


Figura 2. Infecciones por Ransomware a nivel mundial entre Enero de 2015 y Abril de 2016 [3]

En la Figura 2 vemos la evolución del total de infecciones por Ransomware desde principios de 2015. Podemos notar que se mantiene un nivel muy elevado de infecciones y la previsión para el año 2017 es que el Ransomware siga protagonizando la escena del malware a nivel mundial.

1.1 Tipos de Ransomware

Según sus características, podemos diferenciar tres tipos de Ransomware:

- **Ransomware de cifrado**, que incorpora algoritmos avanzados de cifrado. Está diseñado para bloquear los ficheros del sistema y pedir una recompensa por facilitar a la víctima una clave con la que poder recuperar el contenido bloqueado.
- **Ransomware de bloqueo**, que deniega el acceso del usuario al sistema, haciendo imposible acceder al escritorio y a las aplicaciones. En este caso los ficheros no están cifrados pero los atacantes piden una recompensa por devolver el acceso al sistema.
- Un tercer tipo de Ransomware modifica el MBR (Master Boot Record) que es la sección del disco que permite al sistema operativo arrancar con normalidad. El MBR infectado lleva a un proceso de cifrado de la MFT (Master File Table) que contiene los punteros a los ficheros del sistema. Sin acceso a esta tabla, aunque los ficheros no estén cifrados, son prácticamente inaccesibles, ya que su recuperación es extremadamente costosa y no es viable como respuesta ante este cifrado. Para recuperar los datos después de la infección de este tipo de ransomware, es necesario restaurar un "Image backup". Una restauración de la MBR con el disco de instalación y la creación de una nueva MFT no serían suficiente para recuperar el acceso a los ficheros.

Figura 3. Diferencias entre Ransomware de bloqueo y el de cifrado [4]

En la Figura 3 podemos distinguir las características que diferencian al Ransomware de bloqueo del de cifrado.

2.1.1 Ransomware de cifrado (Crypto ransomware)

Normalmente el ransomware usa tanto técnicas de cifrado de clave simétrica[1] como asimétrica[2], comúnmente conocido como cifrado mixto.

En el cifrado simétrico, se usa una única clave para cifrar y descifrar los ficheros. En este tipo de cifrado, hay que tener en cuenta que la clave simétrica debe estar en el equipo, por lo menos, durante el proceso de cifrado. Es imprescindible para el atacante que esta no quede accesible al usuario, pues éste podría realizar el descifrado de los ficheros sin pagar el rescate por ellos.

Para hacer inaccesible la clave simétrica usada para el cifrado, ésta se cifra con la clave asimétrica pública del atacante, siendo éste, el único capaz de descifrarla con su clave privada.

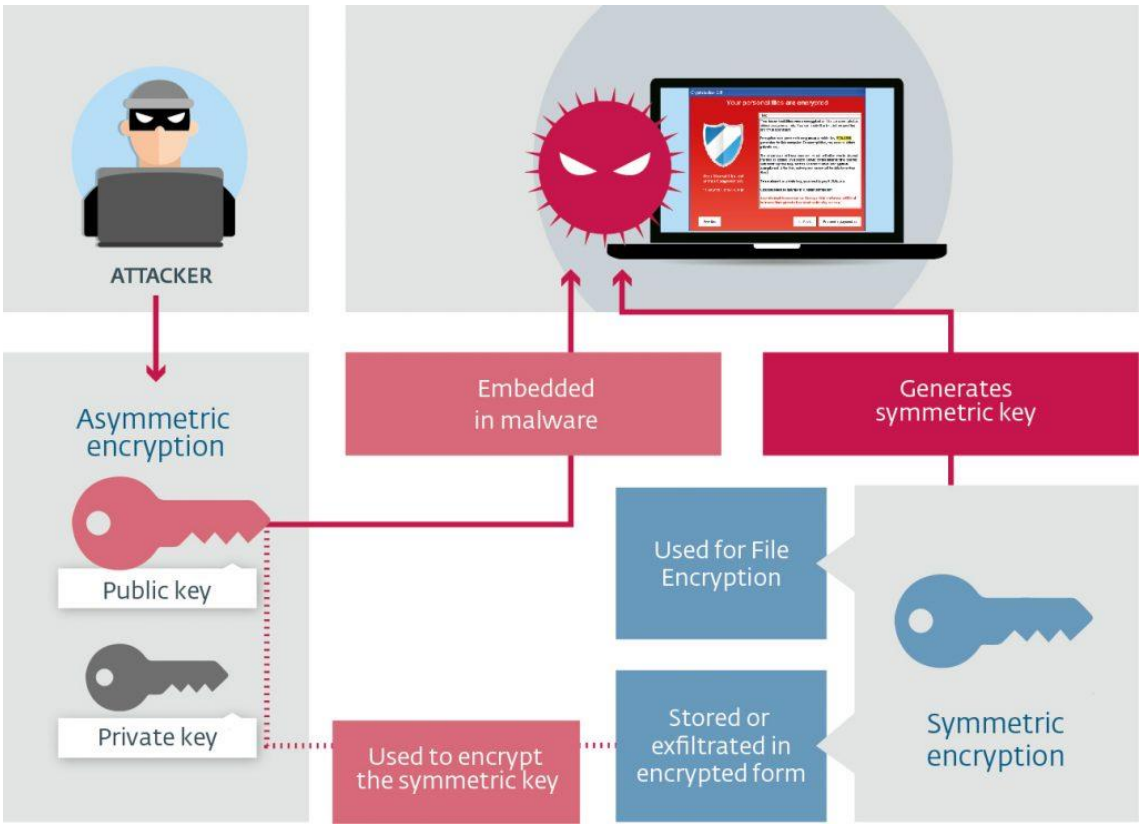


Figura 4. Esquema de encriptación mixta usada en el Ransomware [5]

Este método es usado para obtener las ventajas del cifrado de clave simétrica (principalmente la velocidad) y las ventajas de cifrado asimétrico (robustez).

Para un Ransomware, la velocidad del proceso de cifrado es primordial, para cifrar la mayor cantidad de ficheros en el menor tiempo posible.

---

[1] *Cifrado simétrico. Es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor.*

[2] *Cifrado asimétrico. Es un método criptográfico que consta de 2 claves, una pública y una privada, donde el emisor cifra el mensaje con la clave pública del destinatario para que sólo él pueda descifrarla con su clave privada.*

La generación de la clave simétrica de cifrado se suele realizar de dos formas:

- Las familias de ransomware que requieren descargar la clave necesaria para el cifrado directamente del servidor de control (normalmente AES 256 bits) antes de empezar el cifrado.

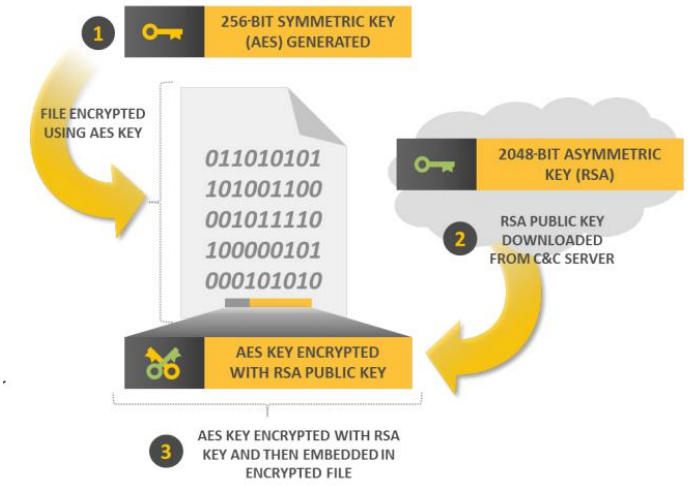


Figura 5. Descargando la clave antes del cifrado [6]

- Las familias de ransomware que la generan localmente en la máquina afectada antes de empezar el proceso y posteriormente la comunican al servidor de C&C[3] (Command and Control).

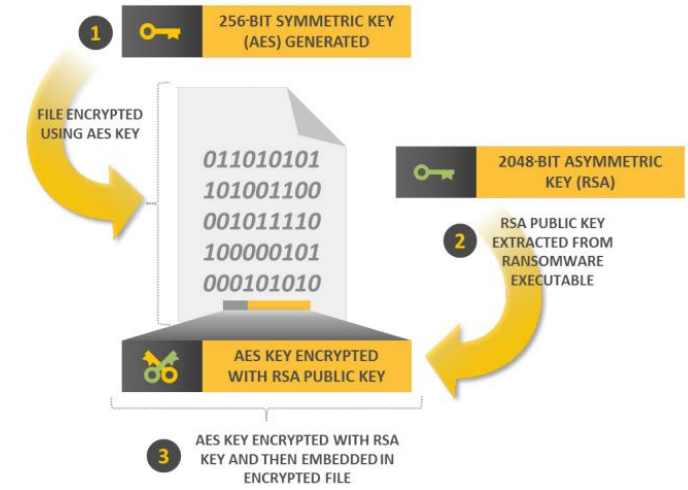


Figura 6. Uso de clave incrustada para el cifrado simétrico [6]

[3] *Command and Control. Son los servidores centralizados desde donde se controla las infecciones, claves de cifrado, estado de los ficheros y estado de la extorsión de los propietarios de los equipos infectados.*

1.2 Afectación y objetivos

En este apartado analizaremos los mecanismos de infección (vectores de ataque) más usados y el alcance de los mismos.

1.2.1 Vectores de ataque

Hay varias vías por las cuales un ransomware puede infectar un ordenador.

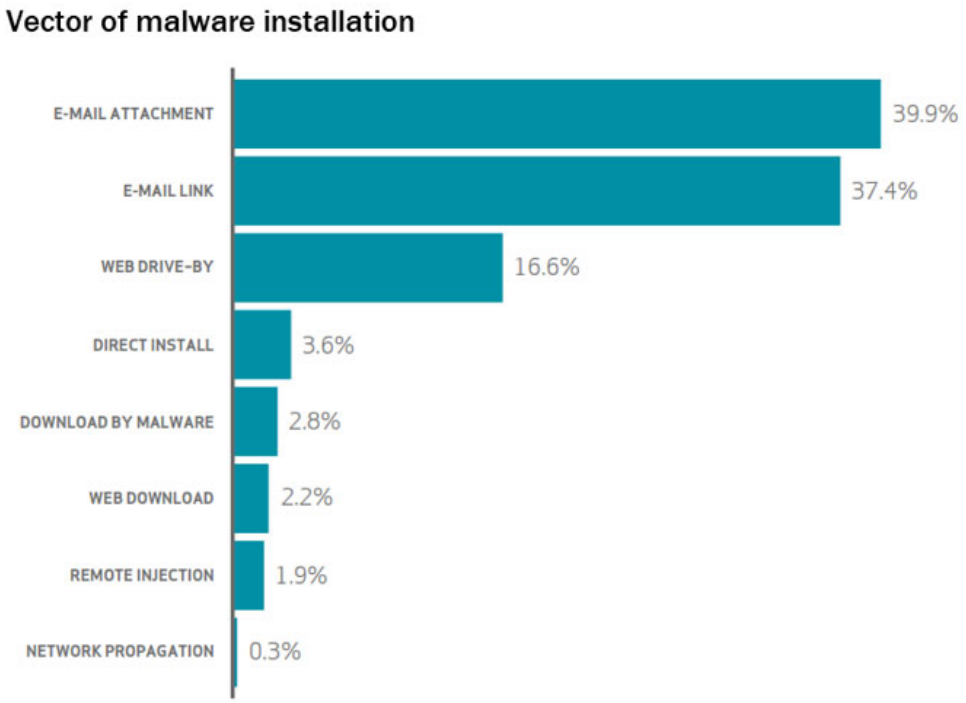


Figura 7. Vectores de instalación de malware más usados en 2015 [7]

En la Figura 7 podemos observar los vectores de infección más usados en el mundo del malware en 2015, también aplicables a la variante de ransomware. En la Figura 8 vemos un ejemplo del flujo de infección

1.2.1.1 Correo malicioso

Uno de los métodos más comunes para difundir ransomware, y malware en general, es a través de correo electrónico spam malicioso. Este correo no deseado se distribuye principalmente mediante botnets, que son las encargadas de gestionar el envío masivo de estos.

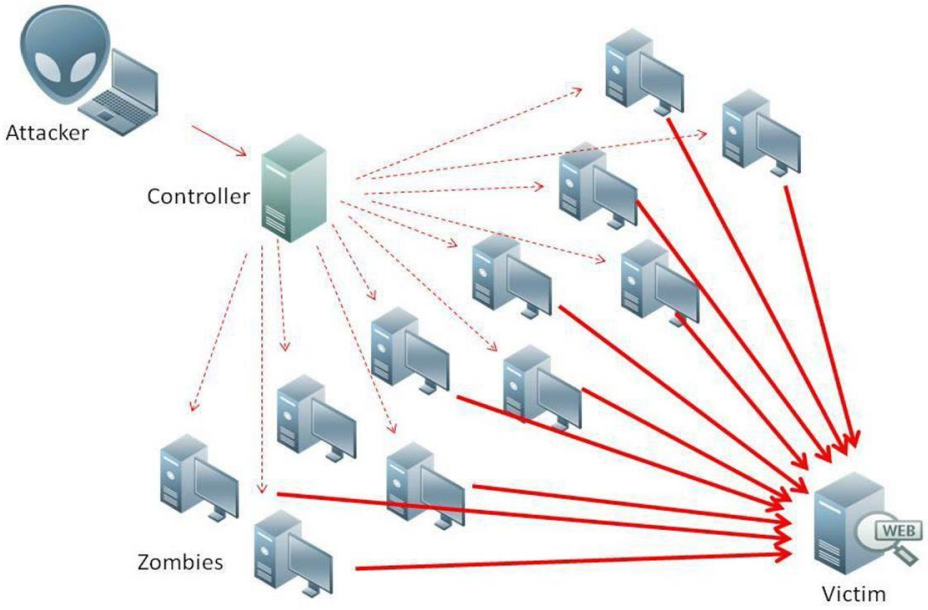


Figura 8. Ejemplo de estructura de botnet centralizada [8]

En la Figura 8 podemos ver la distribución de una botnet centralizada y su administración desde un servidor de command and control (Controller).

	Grum (Tedroo)	***Windigo***	***Cutwail***	***Srizbi***	***Bobax***
Equipos infectados	600.000	10.000	2.000.000	450.000	100.000
E-mails Spam al día (millones)	40.000	35.000	19.000	60.000	27.000

Tabla 1. Ejemplos del alcance de las botnets distribuidoras de spam

En la Tabla 1, podemos ver varios ejemplos de la envergadura que pueden adquirir las redes botnet, para realizar de forma masiva funciones como el envío masivo de SPAM o la DDOS (Denegación de servicio distribuida).

La botnet envía grandes cantidades de correo electrónico no deseado que, mediante tácticas de ingeniería social, intentan engañar a los destinatarios y comprometer sus equipos. La infección puede ocurrir si el usuario lleva a cabo cualquiera de las acciones siguientes:

- Recibe el propio malware adjunto al correo y lo ejecuta (muy improbable debido a que la gran mayoría de soluciones antivirus impedirían su ejecución).
- Recibe y ejecuta un *dropper*[4] adjunto al correo.
- Hace clic en un enlace que apunta a un paquete de exploits[5] que, en última instancia, conduce a que el malware se instale en el ordenador.

El correo no deseado utilizado para distribuir ransomware a menudo se hace pasar por un correo electrónico importante de una organización bien conocida, tal como la siguiente:

- Una notificación por parte de la oficina de correos o de otra compañía de envíos, informando al beneficiario de un paquete o entrega.

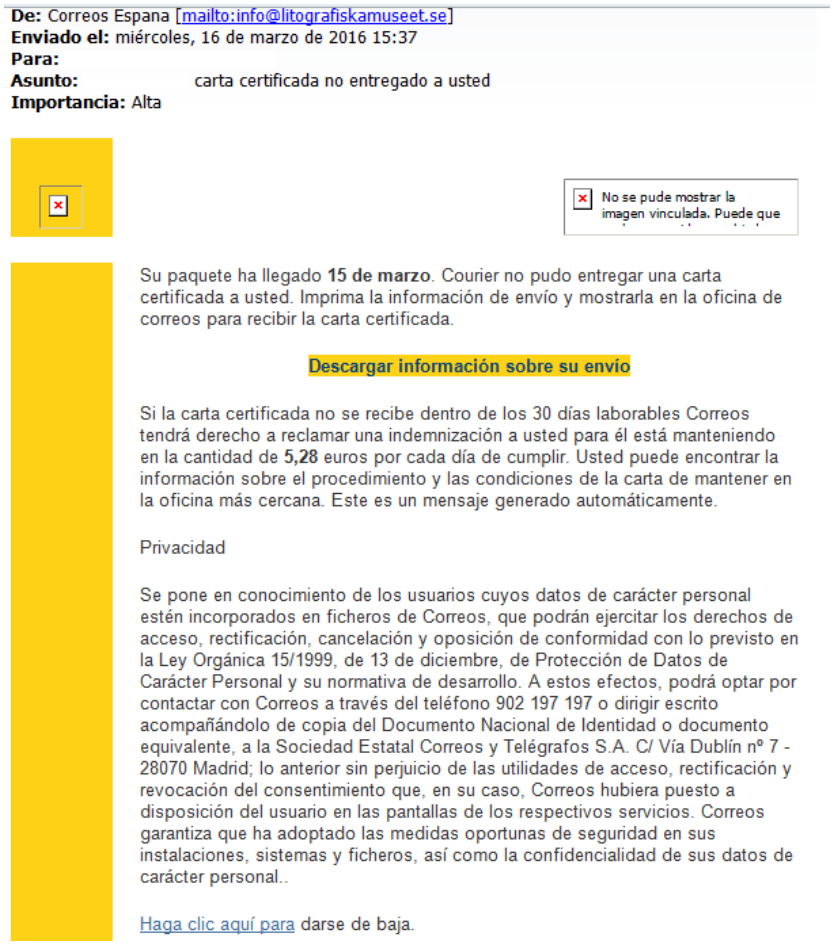


Figura 9. Ejemplo de Phishing de Correos

- Un mensaje de un proveedor de servicios públicos sobre una factura vencida.

[4] *Dropper. Fragmento de código ofuscado para evitar la detección por parte de los sistemas antivirus, que inicia un segundo proceso de descarga (por lo general una macro), que posteriormente descarga e instala el ransomware.*

[5] *Exploit. Fragmento de código utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo.*

- Una alerta sobre la declaración de impuestos del destinatario.
- Programas de remuneración de tarjetas de crédito falsas.
- Facturas correspondientes a bienes y servicios contratados por la mayoría de usuarios (Endesa, Telefónica, etcétera).

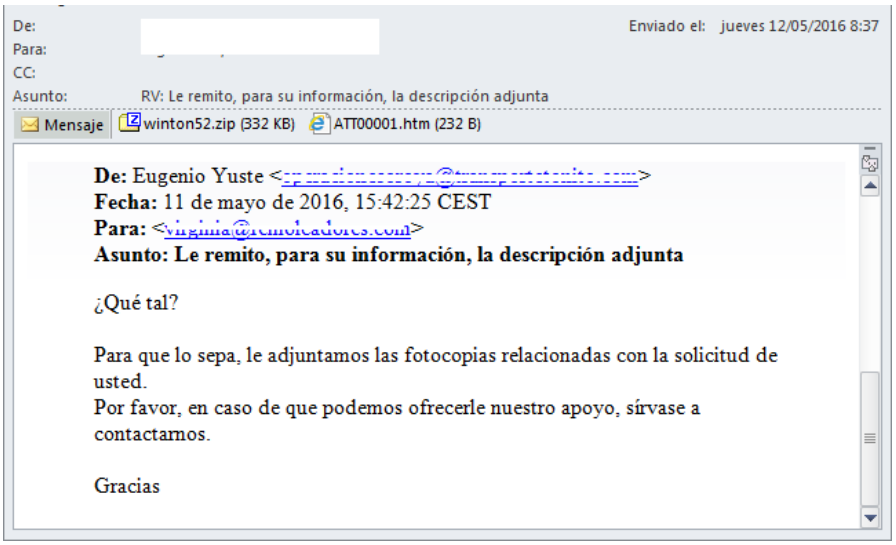


Figura 10. Ejemplo de e-mail malicioso distribuyendo malware adjunto

Cada variación de phishing se basa en el instinto inherente de los usuarios a actuar sobre los mensajes que parecen ser urgentes.

Los atacantes emplean varias técnicas para ayudar a difundir de manera efectiva ransomware a través de correo electrónico.

Durante este año, una gran parte de los atacantes han usado Windows Script Host[6] para vulnerar los filtros de correo electrónico. Una vez el archivo adjunto (una carpeta comprimida que aparentemente contiene un documento ofimático inofensivo) es abierto, el script incrustado se ejecuta y el ransomware se instala en el equipo de la víctima.

[6] *Windows Script Host. Es un motor y entorno de ejecución de scripts para sistemas Windows que interpreta numerosos lenguajes de programación como JS, VBS, WSF y es usado frecuentemente para descargar malware.*

```
var terrifying = druberri.pop().split(">");
var terrifying = "MSXML2.XMLHTTP","WScript:Shell";

var rampart = "WScript:Shell";
var selecting = "MSXML2.XMLHTTP";
var vulture = rampart[druberri.shift()](druberri.shift());

var amalgamation = ".exe";
var promises = druberri.shift();
architectural = "open";
function school(gutter) {
        var appropriations = "%TEMP%/yVrLrAwIvU.exe";
            selecting[architectural]("GET", "htxp://patmagifts.asia/09ujnb76v5?yNVICJbit=nFikKFve", false);

        selecting["send"]();
        var advocacy=("casque" + WScript=="casque" + "Windows Script Host")&&selecting["status"] +"200"&&typ

        if (advocacy) {

            var brings = new logan(ADODB.Stream);
            brings["architectural"]();
            brings.type = 1;
            brings["write"](selecting[("adhering","hebrides","thongs","industries","trends","gymnastic","abs
            brings["position"] = 0;
            brings.saveToFile(appropriations, 2);
            brings.close();
            rampart["Run"](appropriations, 1, true);
        }

}
try{
    school("http://"+"\u0070a\u0074m\u0061g\u0069f\u0074s"+"\u002Ea\u0073i\u0061/\u00309\u0075j\u006Eb\u0037
    http://patmagifts.asia/09ujnb76v5?yNVICJbit=nFikKFve;
```

Figura 11. Ejemplo de dowloader del ransomware Locky WSF donde se muestra la descarga del fichero .exe encargado de cifrar el dispositivo

El correo electrónico, se mantiene como uno de los métodos más efectivos de distribución de malware, debido a que se basa en la ingeniería social; una técnica significativamente más fácil de llevar a cabo que otros mecanismos avanzados.

1.2.1.2 Exploit Kits

Los Exploit Kits (EK) son otro vector de infección prevalente de ransomware. Estos kits aprovechan las vulnerabilidades de software con el fin de instalar malware. Los atacantes comprometen servidores web e inyectan iframes[7] en las páginas web que alojan. Estos iframes redirigen los navegadores hacia los servidores del EK. Los atacantes pueden redirigir al usuario hacia los Exploit Kits de varias maneras:

- Enlaces incluídos en correos maliciosos o posts en redes sociales.
- Malvertising[8].
- Tráfico web redirigido desde servicios de distribución de tráfico (balanceadores).

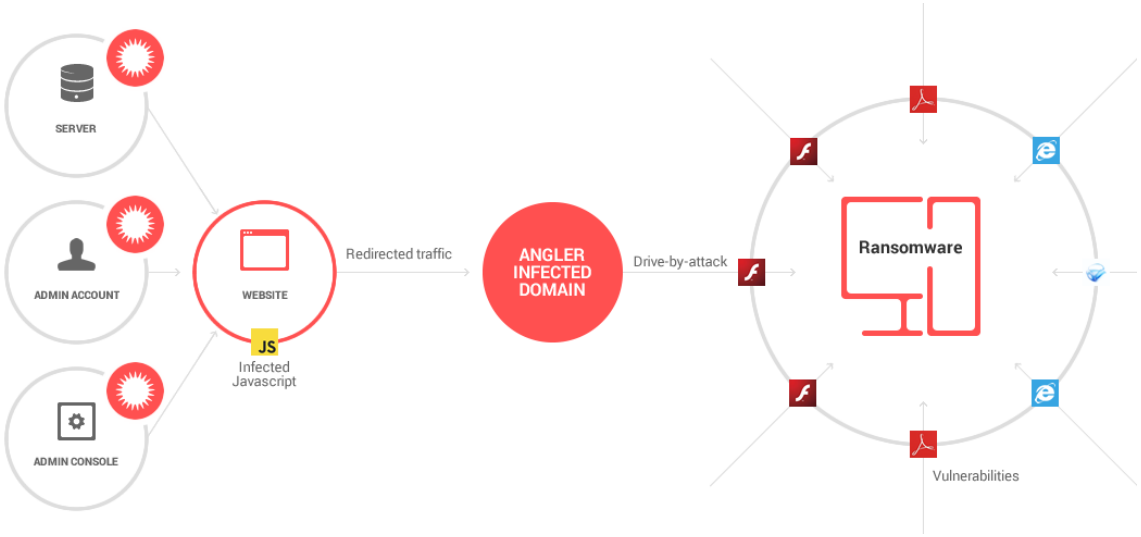


Figura 12. Flujo de infección mediante EK [9]

Los criminales basan su efectividad en el sector de usuarios que usan software sin actualizar y/o parchear. Por fortuna para los atacantes, son una gran parte de los usuarios a nivel global. Por ejemplo, hasta su repentina desaparición en Junio de 2016, Angler EK lideraba el mercado de los Exploit Kits aprovechando vulnerabilidades en Adobe Flash. Los operadores de los EK que usan vulnerabilidades *zero-day*[9], son los que tienen mayor retorno de la inversión. Esto es así, debido a que se trata de vulnerabilidades que aún no tienen solución y, por ende, la afectación en sistemas vulnerables es del 100%.

Los cibercriminales pagan a los operadores de los EK para distribuir su ransomware. Lo normal es que un Exploit Kit se cree en un país, y se venda en otro, sea desde un tercero desde se use para lanzar el ataque hacia un cuarto, y en un quinto donde se localicen los servidores C&C; repartido de esta forma es complicado atribuir a nadie en concreto la actividad maliciosa.

---

[7] *Iframe (Inline Frame). Espacio de navegación anidada que permite incrustar una web dentro de otra.*
[8] *Malvertising. Técnica de distribución de malware basada en la ocultación de código malicioso en anuncios web.*
[9] *Zero-day. Vulnerabilidad no parcheada debido a que aún no es conocida por el fabricante del software afectado.*

Se han convertido en un negocio que permite a un atacante comprar y descargar el kit, crear un ejecutable para explotar vulnerabilidades específicas y además añadir técnicas de evasión para los antivirus, establecer una infraestructura de C&C y empezar a enviar el malware a través de mensajes de spam o páginas web comprometidas. Llegados a este punto, la tarea del atacante se limita a monitorizar esos servidores C&C y recoger las credenciales comprometidas, para después venderlas o hacer uso de ellas.

	Angler	Neutrino	Rig	Magnitude
CryptXXX	✓	✓		
Cerber		✓	✓	✓
Locky			✓	

Tabla 2. Ejemplos de Ransomwares distribuidos por los principales Exploit Kits

En la Tabla 2 podemos ver como un mismo EK es usado indistintamente para distribuir cualquier Ransomware y un ejemplo algunos de los distribuidos actualmente.

2. Definición de objetivos y motivación

RaMON se plantea como una aplicación capaz de activarse en el momento en que el ransomware acaba de cifrar el primer fichero e intenta renombrarlo. En ese punto, realiza las siguientes acciones:

- Detener el proceso encargado de renombrar los ficheros cifrados.
- Renombrar el ejecutable (a extensión .virus) para evitar que se vuelva a ejecutar después de reiniciar el equipo.

Por requisito del cliente, esta aplicación debe ser compatible con los sistemas operativos de los equipos en los que se planee su despliegue y no penalizar en exceso su rendimiento. A tal efecto, se estudiará el parque de OS del entorno y el framework a usar.

2.1 Distribución de los sistemas operativos en entornos empresariales

Dado que la tecnología avanza a gran velocidad, no siempre es fácil mantener actualizado el parque informático de una empresa, y menos cuando se trata de varios miles de equipos. Por esa razón, aún encontramos sistemas ya obsoletos aún en funcionamiento en gran parte de los sistemas empresariales actuales. Por suerte, representan una pequeña parte del conjunto total.

En el siguiente gráfico, podemos ver la cuota de mercado que ocupa Windows XP actualmente en el mercado de sistemas operativos de escritorio y su evolución en el tiempo.

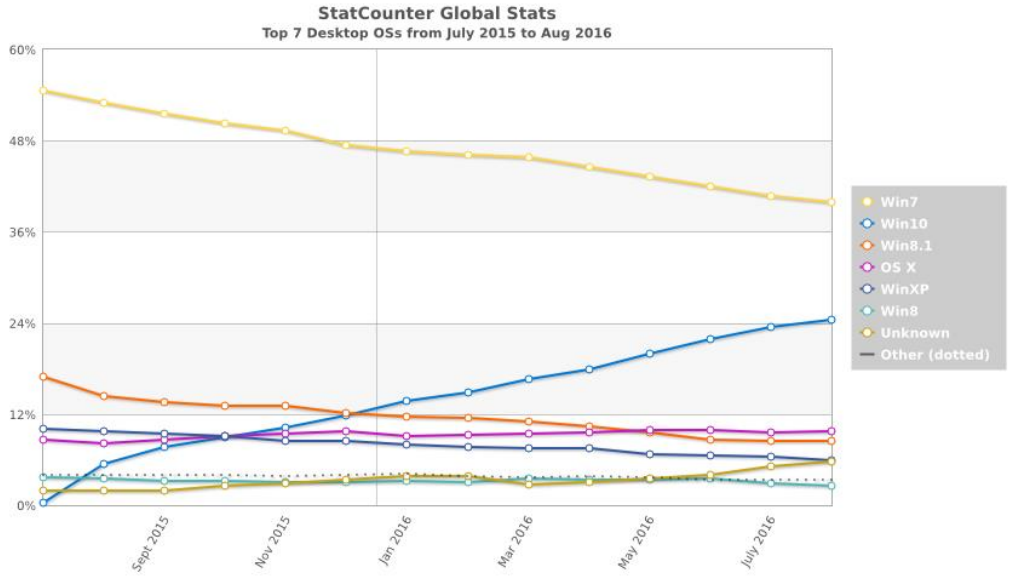


Figura 13. Evolución de la distribución de SO durante el último año [10]

Según estudios de mercado publicados y las estadísticas de StatCounter, la cuota de mercado de Windows XP aún se mantiene cerca del 7%.

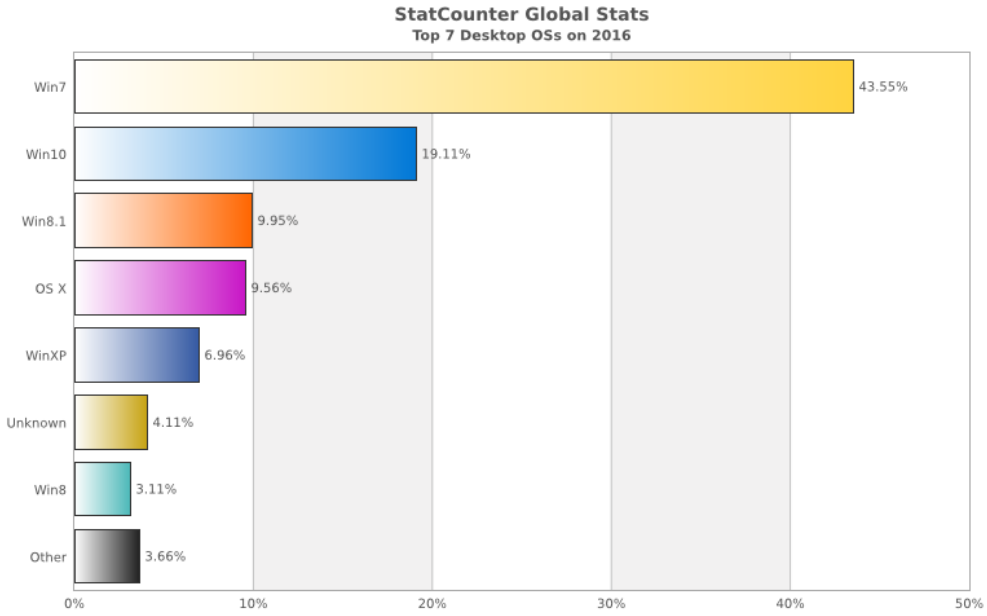


Figura 14. Distribución de la cuota de mercado de sistemas operativos de escritorio durante 2016 [11]

Según los datos recogidos por la empresa Samanage, una importante empresa de servicios informáticos a empresas de Estados Unidos, esta es la distribución de la cuota de mercado en el sector profesional entre los 5 sistemas operativos de escritorio más extendidos actualmente [12].

- Windows 7 Professional 40.3%
- Windows 7 Enterprise 21.4%
- OS X (todas las versiones) 8.1%
- Windows XP Professional 7.3%
- Windows 8.1 Enterprise 4.3%

Podemos observar como en entornos empresariales la cifra de Windows XP es ligeramente superior, debido al coste de la renovación de los parques informáticos a gran escala. Esta evolución, nos permite afirmar que la distribución de este sistema operativo en este tipo de entorno, seguirá reduciéndose de forma gradual hasta su completa desaparición en los próximos años.

RaMON se planteó como una herramienta que debía proteger de las infecciones por ransomware tanto a equipos actuales, como a los ya antiguos Windows XP, debido a que el parque informático para el que se desarrolló, todavía cuenta con una significante cantidad (alrededor del 30%) de equipos con este sistema operativo. Este es solo un pequeño ejemplo que confirma que actualmente tenemos infraestructuras críticas que siguen usando software obsoleto y vulnerable.

2.2 Otras herramientas Anti-Ransomware en el mercado

Vamos a hacer un breve análisis de las principales características de las aplicaciones Anti-Ransomware que podemos encontrar actualmente, con el fin de dar sentido a nuestra solución.

2.2.1 Anti-Ransomware de Malwarebytes

Herramienta descontinuada y añadida a Malwarebytes Anti-Malware. Se basa en monitorización y análisis heurístico.

Actualmente la herramienta, en su versión gratuita, no dispone de protección en tiempo real (la necesaria para el tipo de detección que necesitamos) así que no cubre nuestra necesidad, tal y como podemos observar en la Figura 15.

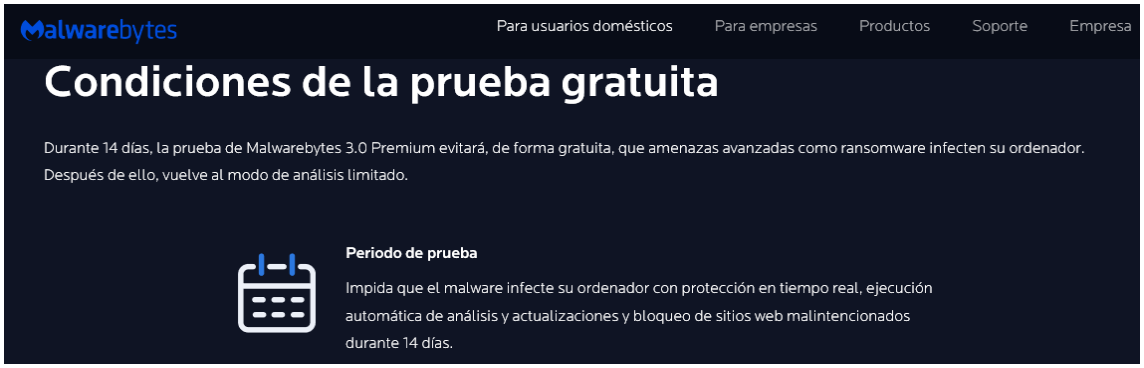


Figura 15. Protección en tiempo real desactivada en versión gratuita MBAM [13]

En la Figura 15 podemos ver que en la web oficial de la aplicación indican que la protección en tiempo real sólo está disponible en la versión Premium de la aplicación.

2.2.2 HitmanPro Alert – de pago

Herramienta anti malware genérica a la que se ha añadido un módulo específico de análisis de amenazas de tipo ransomware. Se basa en un driver que monitoriza el comportamiento de los ficheros (llamadas a funciones susceptibles de un uso malicioso, creación y rename de ficheros, y modificación de claves de registro) y ofrece una versión gratuita de 30 días de validez.

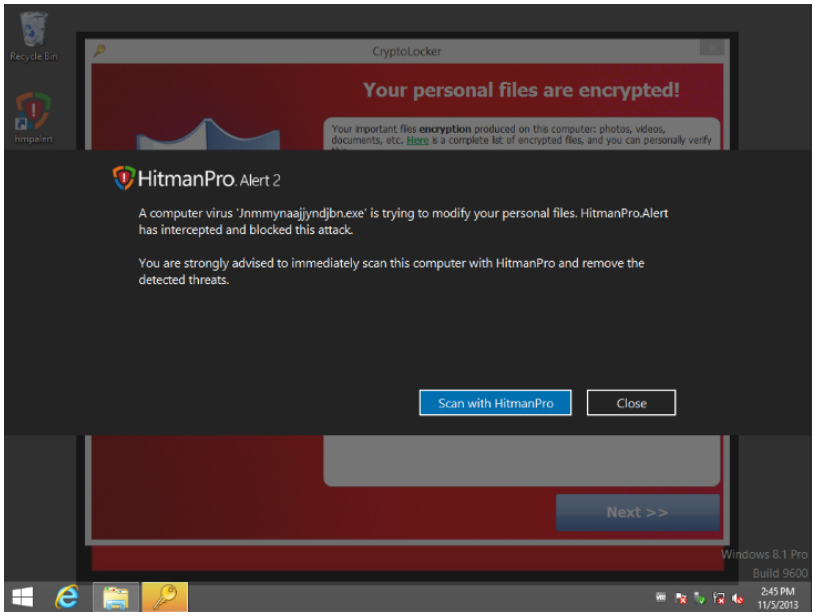


Figura 16. Detección de Ransomware de HitmanPro Alert [14]

2.2.3 Cryptostalker - Linux

Cryptostalker es una herramienta enfocada a la detección y bloqueo de ransomware en entornos Linux. El funcionamiento es, conceptualmente, parecido al de RaMON; monitoriza el sistema de ficheros en busca de un indicio de cifrado y, en caso de encontrarlo, busca al posible responsable de este.

Figura 17. CryptoStalker [15]

2.2.4 CryptoPrevent – Solo CryptoLocker - GPO

Paquete de reglas GPO capaces de proteger el sistema mediante la gestión adecuada de permisos (tanto de usuarios y grupos como de rutas y tipos de fichero).

Según su página web, la herramienta lleva a cabo un análisis de comportamiento (como tácticas de infección y puntos de ejecución) y, según indican, detecta todo tipo de malware que se comporte como CryptoLocker (como CryptoWall o CryptoFortress).

Por este motivo, el hecho de no abarcar otras familias de ransomware ha sido determinante a la hora de descartarla como herramienta de amplio espectro.

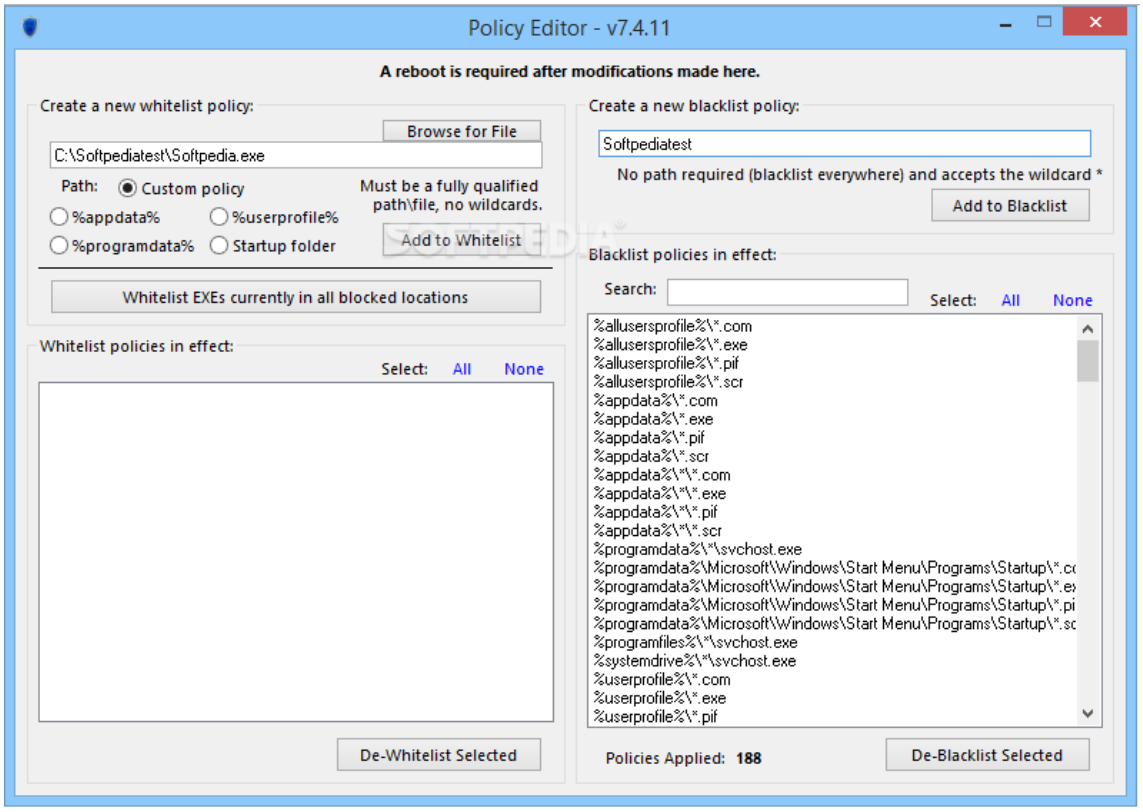


Figura 18. Módulo BlackList/WhiteList[10] de CryptoPrevent [16]

Como podemos ver en la Figura 16, CryptoPrevent ofrece un módulo de gestión de listas negras y listas blancas que permiten aplicar políticas de “Allow by Default” o “Disallow by Default” muy útiles para el control de permisos de ejecución.

[10] Whitelist/Blacklist. Lista de elementos permitidos/no permitidos en un determinado proceso.

2.2.5 AntiRansom v3 - Python - SecurityByDefault

Monitor del sistema de ficheros, programado inicialmente en Perl y posteriormente llevado a Python, es una herramienta muy parecida a nuestra solución pero que presenta dos inconvenientes que no la hacían válida para nuestra necesidad:

- Requiere tener instalado Python en la máquina a monitorizar.
- Utiliza *canary files*[11] para detectar el proceso de cifrado en una carpeta señuelo, a modo de HoneyPot. A nuestro entender, no es una solución fiable, ya que no podemos confirmar a priori por qué carpeta va a empezar a cifrar el ransomware.

Figura 19. Interfaz de la herramienta Anti Ransom v3 [17]

En la imagen de la derecha, podemos observar cómo antes de detener el proceso pide al usuario su confirmación. Cabe destacar que esta aplicación, a diferencia de RaMON, pausa el proceso para dejar a elección del usuario la acción de finalizarlo.
Esta opción se valoró inicialmente para RaMON, aunque se descartó debido a varias incompatibilidades. Una de las principales, son las familias de ransomware que generan un proceso por cada fichero que cifran; mostrar una ventana al usuario por cada proceso potencialmente malicioso suponía perder la estabilidad de la aplicación.
Después de valorarlo, decidimos que la posibilidad de que un usuario validara y no detuviera un proceso malicioso pensando que es legítimo, sumado a la inestabilidad que esta característica podría aportar a RaMON, era suficiente para no implementarla. Debemos recordar que un falso positivo de la aplicación supone la finalización de un proceso no crítico del usuario y una reconfiguración posterior de las interfaces de red, mientras que una decisión errónea del usuario puede llevar a la pérdida de todos los ficheros de su equipo y de los ubicados en las unidades de red a las que tenga acceso. Creemos que es un riesgo demasiado elevado para asumirlo, puesto que el usuario al que va dirigido no tiene porqué saber diferenciar un proceso malicioso de uno legítimo (la mayoría de malware usa nombres de aplicaciones conocidas para pasar desapercibidos. Por ejemplo: Dropbox.exe).

[11] *Canary Files. Archivos basura que tienen como objetivo avisar si detecta un proceso de cifrado.*

2.2.6 AppLocker - GPO - Microsoft

Applocker se basa en la aplicación de políticas de grupo tanto para usuarios y grupos como para aplicaciones. Para la gestión de aplicaciones permitidas, hace uso de *whitelist* y *blacklist*, tal y como hemos visto en otras aplicaciones del sector.

Aporta la posibilidad de crear reglas y directivas, así como excepciones a las mismas.

Su funcionalidad se basa en PowerShell, y esto solo lo hace compatible con versiones de Windows a partir de Windows 7 en adelante.

En la tabla siguiente se compara AppLocker con las directivas de restricción de software.

Característica	**Directivas de restricción de software**	**AppLocker**
Ámbito de regla	Todos los usuarios	Usuario o grupo concreto
Condiciones de regla proporcionadas	Hash de archivo, ruta de acceso, certificado, ruta de acceso del Registro y zona de Internet	Hash de archivo, ruta de acceso y publicador
Tipos de regla proporcionados	Definido por los niveles de seguridad: • No permitido • Usuario básico • Sin restricciones	Permiso y denegación
Acción de regla predeterminada	Sin restricciones	Denegación implícita
Modo de solo auditoría	No	Si
Asistente para crear varias reglas de una vez	No	Si
Importación o exportación de directiva	No	Si
Colección de reglas	No	Si
Compatibilidad con Windows PowerShell	No	Si
Mensajes de error personalizados	No	Si

Tabla 3. Diferencias entre AppLocker y las directivas de restricción de software

2.2.7 BDAntiRansomware de BitDefender

BitDefender Anti-Ransomware Vaccine es un programa que protege de crypto-ransomware. Monitoriza los archivos del sistema buscando actividades similares a parásitos que cifran los archivos y luego piden dinero para su recuperación.

Solo protege de Locky, Tesla Crypt y CryptoLocker, hecho que lo hacía inadecuado a nuestras necesidades.

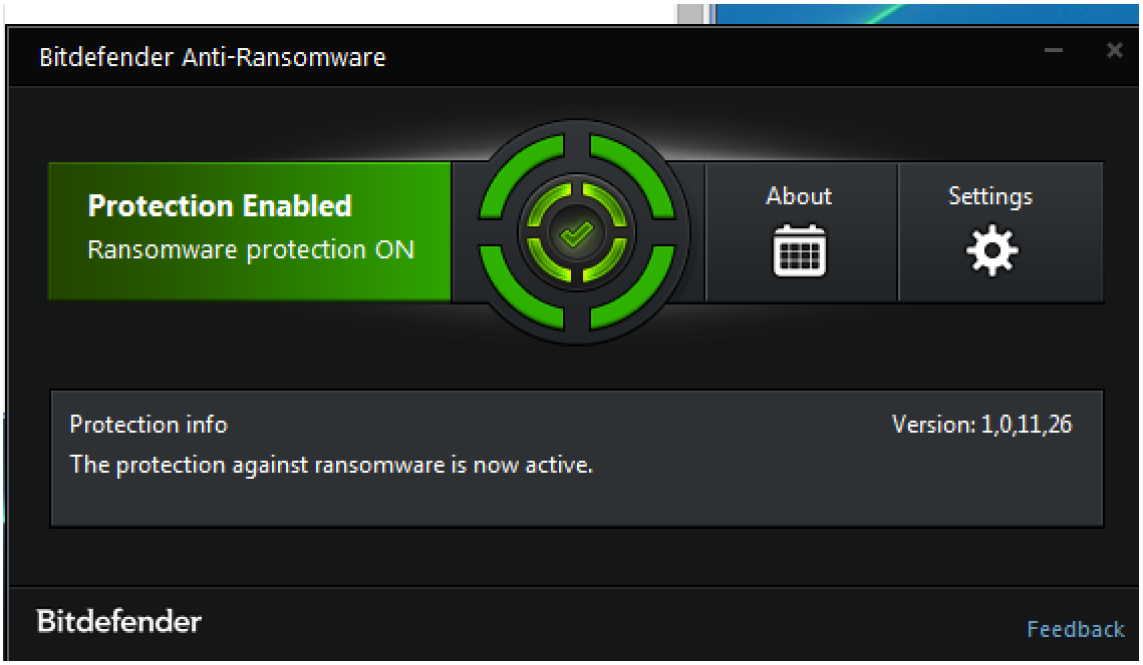


Figura 20. Bitdefender Anti-Ransomware[18]

2.2.8 PROTEIN – PROTEct your INformation

Monitorea activamente los archivos de un repositorio en busca de la actividad de ransomwares.

- Identifica archivos conocidos (lista blanca), amenazas potenciales (lista negra) así como archivos desconocidos para su posterior procesamiento.
- Registra cada acción de creación de archivos en el servidor.
- Permite enviar alertas por correo electrónico a un administrador cuando se detecta un nuevo ransomware.
- Permite alertar directamente en pantalla a un usuario cuando se detecta un nuevo ransomware.
- Desactiva el usuario de dominio, impidiendo el acceso al repositorio por el ransomware o a otros objetos críticos del sistema.
- Desactiva la NIC del usuario afectado por el ransomware para bloquear el acceso a la red.

Figura 21. Ayuda de la herramienta PROTEIN [19]

El funcionamiento de PROTEIN es muy similar al de RaMON. Aunque su desarrollo se inició más tarde que RaMON, vimos en PROTEIN el objetivo que buscábamos en nuestra aplicación.

El punto más importante de diferenciación entre ambas, es el lenguaje en el que han sido desarrolladas, ya que PROTEIN requiere tener instalado Python en el sistema para poder funcionar.

3. Desarrollo

En este apartado detallaremos los pasos seguidos durante el proceso de arquitectura y diseño de la aplicación.

3.1 Posibles modelos

Las posibilidades para la creación de esta herramienta que se plantearon en un principio fueron las siguientes:

- Hooks
- Minifilter driver
- Aplicación de consola

3.1.1 Hooks

Un hook es una función que permite interceptar: mensajes (hooks de windows), eventos (Driver hooks) o, en el caso que nos ocupa, llamadas a función de sistema (API hooks).

Con un API hook, podemos interceptar esas llamadas a funciones de sistema que nosotros consideremos sospechosas, como por ejemplo RegOpenKeyEx, RegEnumValue, que se encargan de abrir una clave de registro de windows y obtener su valor respectivamente.

La primera idea fue hookear las llamadas a funciones que suponían un indicador de actividad relacionada con el cifrado malicioso. Esta opción conllevaba el problema que, debido a que no conocemos a priori el nombre del ejecutable malicioso, se debería crear un hook en Explorer que fuera haciendo a su vez hook de todo proceso que se levantara en el sistema, pudiendo ocasionar problemas en el mismo.

Esta opción se descartó por la elevada probabilidad de falsos positivos que implicaba no conocer el proceso del que queremos capturar las llamadas.

3.1.2 Minifilter driver

Un minifilter driver es un controlador capaz de filtrar ciertos tipos concretos de operaciones I/O en el sistema. La idea era usarlo para filtrar las operaciones que fueran indicador de un proceso de cifrado malicioso.

Esta idea fue descartada debido a la complejidad que suponía la programación de un driver de sistema y la peligrosidad e inestabilidad que puede provocar el no hacerse de forma adecuada.

3.1.3 Aplicación de consola

Una aplicación de consola nos permitía una gestión y parametrización mediante una interfaz textual. Podía establecerse en el arranque del sistema e interactuar directamente y a más alto nivel con los procesos del sistema y las operaciones realizadas en el sistema de ficheros que nosotros consideramos indicador de cifrado malicioso.

3.2 Modelo elegido

La elección final, debido a las ventajas mencionadas, fue aplicación de consola. De esta manera, basándonos en una *blacklist* de extensiones de fichero (aprovechando el hecho de que la mayoría de ellas son usadas únicamente en procesos de cifrado malicioso y que, adicionalmente, en muchos casos nos permiten identificar la familia de ransomware), podríamos detectar cuando un ransomware está empezando a cifrar la unidad de un usuario comprometido.

Debido a que queríamos monitorear el sistema de ficheros, estudiamos los diferentes lenguajes con los que podíamos hacer la aplicación.

.NET y C++ incorporan un objeto que interactúa con el sistema y que podíamos usar como motor de nuestra herramienta.

- **.NET** nos ofrece una programación más cómoda y a más alto nivel. Por contra, el empaquetado final y el consumo de recursos son más elevados.
- **C++** es un lenguaje de más bajo nivel que, a nivel de interacción con el sistema operativo, puede ser difícil de programar. Este nivel nos ofrece una gran flexibilidad a la hora de acceder a las funciones y valores del sistema.

Debido a que el peso de la aplicación final y el consumo de recursos eran dos factores clave en nuestro diseño, sumado a que C++ nos proporciona una capa de más bajo nivel, elegimos C++ como lenguaje para el desarrollo de RaMON.

En la siguiente imagen, se representa el diagrama de flujo del funcionamiento de la aplicación.

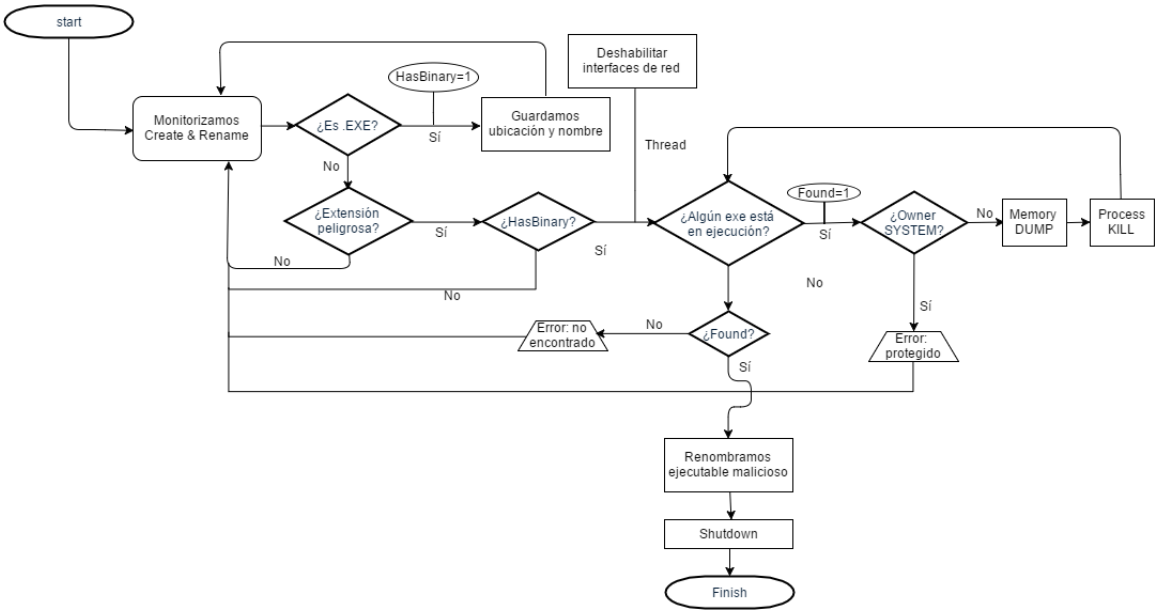


Figura 22. Diagrama de flujo de la aplicación

En el Anexo 2, podemos ver el diagrama en tamaño ampliado para apreciar los detalles.

4. Metodología

En este apartado explicaremos cómo ha sido estructurada la aplicación, sus partes principales y el funcionamiento de la misma.

4.1 Estructura de la aplicación

En este apartado haremos referencia a la configuración, motor y base de datos que forman la base del mecanismo de RaMON.

4.1.1 Motor

Una vez elegido el formato de la aplicación, procedemos a desarrollar el motor de RaMON, el objeto que monitoriza el sistema de ficheros. Para esa función, usamos el objeto FileSystemWatcher de C++, capaz de detectar cambios en el sistema de ficheros (create, delete and rename).

El objeto FileSystemWatcher escucha las notificaciones de cambio del sistema de archivos y genera eventos cuando cambia un directorio o un archivo de un directorio.

Constructores

Nombre	Descripción
FileSystemWatcher()	Inicializa una nueva instancia de la clase FileSystemWatcher.
FileSystemWatcher(String^)	Inicializa una nueva instancia de la clase FileSystemWatcher, dado el directorio especificado que se va a supervisar.
FileSystemWatcher(String^, String^)	Inicializa una nueva instancia de la clase FileSystemWatcher, dado el directorio especificado y el tipo de archivos que se van a supervisar.

Tabla 4. Constructores del objeto FileSystemWatcher

Propiedades usadas

Nombre	Descripción
Path	Obtiene o establece la ruta de acceso del directorio que se va a inspeccionar.
NotifyFilter	Obtiene o establece el tipo de cambios que se van a inspeccionar.
Filter	Obtiene o establece la cadena de filtro utilizada para determinar qué archivos se supervisan en un directorio.
Events	Obtiene la lista de controladores de eventos que se adjuntará a este Component.

Tabla 5. Propiedades del objeto FileSystemWatcher usadas en el desarrollo de RaMON

Parametrizamos el objeto para que monitorice la unidad que queremos y activamos los triggers de creación y modificación de ficheros.

```
/* Función que crea los monitores */
static void CreateMonitor()
{
    FileSystemWatcher^ watcher = gcnew FileSystemWatcher;
    Console::WriteLine( "Indique el nombre de la unidad: [C]" );
    System::String^ Path = Console::ReadLine();
    if ( Path != "" ){
        Path = Path->Concat(Path,":\\");
        watcher->Path = Path;
    }else
        watcher->Path = "C:\\";
    watcher->NotifyFilter = static_cast<NotifyFilters>(NotifyFilters::LastAccess |
        NotifyFilters::LastWrite | NotifyFilters::FileName | NotifyFilters::DirectoryName);
    watcher->IncludeSubdirectories = true;
    watcher->Created += gcnew FileSystemEventHandler( Watcher::OnChanged );
    watcher->Renamed += gcnew RenamedEventHandler( Watcher::OnRenamed );
    watcher->EnableRaisingEvents = true;
}
```

Figura 23. Parametrización del objeto FileSystemWatcher en RaMON

4.1.2 Configuración

Para una gestión cómoda de los parámetros que usa RaMON durante su monitorización, hemos creado un fichero *settings.conf* donde deberemos introducir:

- URL_EXTENSIONS: Enlace de descarga de la *blacklist*.
- TIME_TO_UPDATE: Intervalo de tiempo entre cada actualización (en minutos).

```
settings.conf
URL_EXTENSIONS=https://docs.google.com/document/d/1LWwFv7e_rFwdKfTb_OQzmUh_11O2z2IpNJdxPDbZivg/export?format=txt
TIME_TO_UPDATE=10
```

Figura 24. Fichero de configuración de RaMON

La función que realiza la lectura de la configuración es ReadConfig().

4.1.3 BlackList

Para determinar si un fichero ha sido cifrado y así poder realizar las tareas de búsqueda y detención de un posible proceso malicioso, utilizamos una *Blacklist*.

Una *blacklist* es un listado de ítems (dominios, IP's, URL's, directorios,...) que utiliza una aplicación o servicio para determinar que contenido no está permitido. Estas son usadas, por ejemplo, por software para el control de permisos, control de navegación, control parental o firewalls.

Para nuestra aplicación hemos desarrollado una *blacklist* de extensiones peligrosas, que usamos como indicador de que un fichero ha sido cifrado. De esta manera nuestra *blacklist* se convierte en nuestra base de datos de indicadores de compromiso. Así, si nuestro sistema de monitorización detecta una de esas extensiones en un fichero, activamos de inmediato los mecanismos de defensa.

Nuestra *blacklist* se encuentra en el mismo directorio de la aplicación, con el nombre *extensions.txt*.

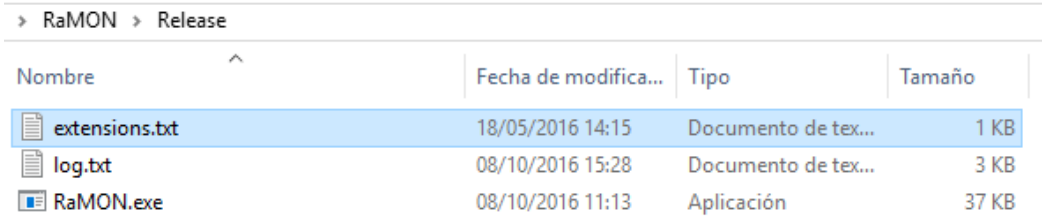


Figura 25. Blacklist de extensiones peligrosas

El principal reto de esta aplicación, junto a la capacidad de detectar y bloquear las amenazas, ha sido plantear una solución al hecho de que continuamente están apareciendo nuevos ransomwares y nuevas variantes de los ya existentes y debemos tener un repositorio centralizado, actualizado y accesible para nuestra base de datos de IOC's.

Como origen de los datos para esta prueba de concepto, usamos un documento que está siendo continuamente actualizado con las nuevas amenazas y de allí obtenemos las extensiones.

https://docs.google.com/document/d/1LWwFv7e_rFwdKfTb_OQzmUh_11O2z2IpNJdxPDbZivg/export?format=txt

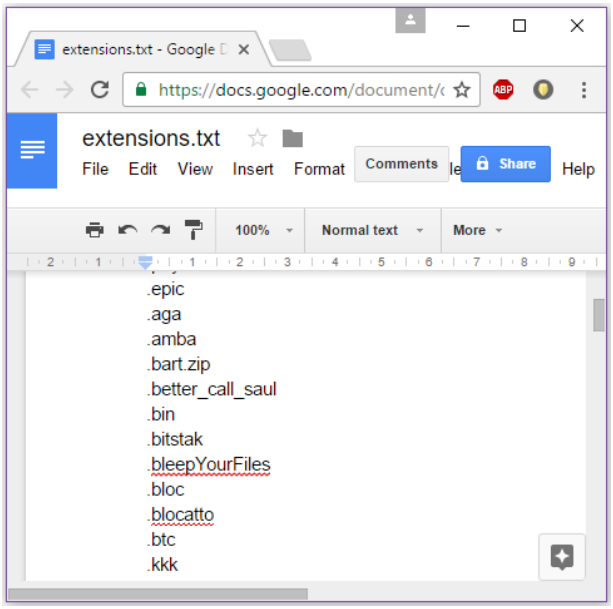


Figura 26. Listado de extensiones maliciosas

Tal y como vemos en la Figura 25, el formato de los registros de la *blacklist* es del tipo "*.extension_peligrosa"* introducidos a registro por línea.

Una vez revisadas y añadidas a nuestro fichero, añadimos la URL al fichero de configuración y podemos iniciar la monitorización. Por defecto estará configurado con nuestro fichero de *blacklist*. La función que se encarga de la descarga y sustitución de la *blacklist* es *UpdateBlackList().*

```
LPCTSTR Url = _T((char*)Marshal::StringToHGlobalAnsi(urlUpdateExtensions).ToPointer()), File = _T("extensions.tmp");
//Descagamos el nuevo fichero de nuestro repositorio
hr = URLDownloadToFile (0, Url, File, 0, 0);
switch (hr)
{
    case S_OK: //Descarga correcta
        sprintf_s(strLog,256,"Descarga extensions.dat correcta\n");
        Log(strLog);
        if(!FileExists("extensions.dat")){ //No existe fichero extensions.dat
            rename("extensions.tmp", "extensions.dat");
        }else{
            if(!watcher){ //Descarga inicial (watcher no activado)
                DeleteFile(TEXT("extensions.dat"));
                rename("extensions.tmp", "extensions.dat");
                DeleteFile(TEXT("extensions.tmp"));
            }else{ //Actualización del fichero extensions.dat (watcher activado)
                //Desactivamos la monitorización
                watcher->EnableRaisingEvents = false;
                watcher->Created -= gcnew FileSystemEventHandler( Watcher::OnCreated );
                watcher->Renamed -= gcnew RenamedEventHandler( Watcher::OnRenamed );

                //Sustituimos el nuevo fichero blacklist por el antiguo
                DeleteFile(TEXT("extensions.dat"));
                rename("extensions.tmp", "extensions.dat");
                DeleteFile(TEXT("extensions.tmp"));

                //Volvemos a activar la monitorización
                watcher->Created += gcnew FileSystemEventHandler( Watcher::OnCreated );
                watcher->Renamed += gcnew RenamedEventHandler( Watcher::OnRenamed );
                watcher->EnableRaisingEvents = true;
            }
```

Figura 27. Descarga de la nueva blacklist (extensions.dat)

4.2 Monitorización

Durante la fase de monitorización buscamos capturar el nombre y directorio de los ejecutables que se crean en el sistema y, paralelamente, detectar el indicador de cifrado malicioso (en nuestro caso, el uso de extensiones de fichero que tenemos en nuestra *blacklist*).

4.2.1 Eventos *OnCreated* y *OnRenamed*

Para llevar a cabo tanto la detección de ficheros ejecutables susceptibles de ser el malware como los ficheros cifrados, usamos los eventos Watcher::OnCreated y Watcher::OnRenamed

```
/* Función que se activa cuando detecta que se crea un fichero */
static void OnCreated( Object^ /*source*/, FileSystemEventArgs^ e )
{
    CheckFile(e->Name, e->FullPath);
}

/* Función que se activa cuando detecta que se renombra un fichero */
static void OnRenamed( Object^ /*source*/, RenamedEventArgs^ e )
{
    CheckFile(e->Name, e->FullPath);
}
```

Figura 28. Eventos OnCreated y OnRenamed

Watcher::OnCreated: Se ejecuta cada vez que se crea un nuevo fichero en el sistema monitorizado.

Watcher::OnRenamed: Se ejecuta cada vez que se detecta la acción de renombrar un fichero existente.

Con el fin de realizar la detección tanto de los ficheros ejecutables como de los cifrados en los dos tipos de eventos, llamamos a la función *CheckFile* en los dos casos.

```
Funcion CheckFile(fichero)
    Si (fichero.extension == ".exe") Entonces
        Si fichero.path <> "RECYCLER" y fichero.path <> "System Volume Information" Entonces
            Mostrar "Binario detectado"
            Si (index==SIZE) Entonces
                index = 0;
            FinSi
            lastBinaries[index,0] = fichero.nombre;
            lastBinaries[index,1] = fichero.path;
            index++;
        FinSi
    Sino
        SearchCryptoExtension(fichero.nombre);
    FinSi
FinFuncion
```

Figura 29. Función CheckFile que busca ficheros .EXE

Si detectamos que se trata de un fichero ejecutable, comprobamos que no se encuentre ni en la papelera ni en la carpeta System Volume Information con el fin de evitar falsos positivos. En caso negativo, guardaremos el nombre del fichero y su ubicación con el fin de poder localizarlo *a posteriori* en caso de ser considerado el posible origen de un cifrado malicioso.

Una de las técnicas más usadas en el mundo del malware para hacer pasar desapercibido el proceso es usar el nombre de un proceso protegido de sistema.

Este sencillo detalle, puede hacer que una aplicación no pueda finalizar el proceso interpretando que se trata del proceso protegido de sistema.

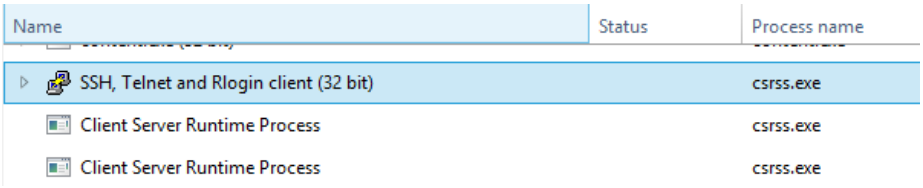


Figura 30. Ejemplo de un uso malicioso de un proceso protegido de sistema

Durante las pruebas observamos que internamente el sistema añadía a nuestro proceso un [1] al final del nombre, por lo que pasaba a llamarse csrss[1].exe.

```
if(lastBinaryName->Contains("[1]")){//si el binario tiene nombre de proceso protegido se registrara como proceso[1].exe (W7)
    sprintf_s(strLog,256,"Uso de nombre protegido: %s\n",lastBinaryName);
    lastBinaryName = lastBinaryName->Substring(0,lastBinaryName->LastIndexOf("["));
    lastBinaryName = lastBinaryName->Concat(lastBinaryName,".exe");
    lastBinaryPath = ePath->Substring(0,ePath->LastIndexOf("["));
    lastBinaryPath = lastBinaryPath->Concat(lastBinaryPath,".exe");
}else lastBinaryPath = ePath;
```

Figura 31. Mecanismo de control para evitar uso de nombres protegidos

Para evitar que se pudiera saltar la protección de la herramienta con este mecanismo, añadimos un control y posteriormente registramos el nombre y el directorio donde se encuentra el binario.

Se almacenan los 5 últimos ficheros ejecutables, que serán los que se buscarán para ser finalizados en caso de detección de un proceso de cifrado malicioso.

4.3 Detectar de extensiones peligrosas (IOC)

Si el fichero analizado no es un ejecutable, lanzamos la función *SearchCryptoExtension*, que valida que la extensión de un fichero no se encuentre en la *blacklist* con la que trabaja la aplicación.

```
/* Función encargada de detectar extensiones maliciosas */
static void SearchCryptoExtension(System::String^ fileName){

    String^ ext; //Extension de la blacklist
    String^ lastBinaryPath; //Path del binario
    int res = 0; //Flag si ha matado algun proceso
    char strLog[256]; //String para el log
    StreamReader^ din = File::OpenText("extensions.dat");
```

Figura 32. Función SearchCryptoExtension

Comprobamos si la extensión del fichero figura en nuestra *blacklist* y, si es así, procedemos a lanzar un nuevo *thread* que se encargará de deshabilitar las interfaces de red.

```
/* Tumbamos las interficies de red */
ThreadStart^ start = gcnew ThreadStart(w, &Watcher::DisableNetInterfaces);
Thread^ t = gcnew Thread(start);
t->Start();
```

Figura 33. Creamos un nuevo thread para deshabilitar las interfaces de red

4.4 Deshabilitar interfaces de red

Este paso es el primero para evitar una posible propagación por la red a la que se encuentra conectada el usuario. Es muy común que este tipo de malware escanee las unidades de red, en busca de equipos externos a los que tenga acceso para poder extenderse e infectar el mayor número de equipos. Es vital evitar ese salto, por esa razón, ésta es la primera de las tareas que se llevan a cabo en el momento de determinar que existe una posible infección. Se ha incluido en un nuevo hilo de procesamiento para reducir al mínimo el tiempo entre la detección y la detención del proceso encargado de cifrado.

```
/* Función encargada de deshabilitar las interfaces de red */
void DisableNetInterfaces(){
    system("for /F tokens^=2^ delims^=^\" %a in ('netsh int ip show config')
        do netsh interface ip set address name=\"%a\" static 169.254.0.100 255.0.0.0 169.254.0.0 1");
    Log("Deshabilitamos las interfaces de red\n");
}
```

Figura 34. Función DisableNetworkInterfaces encargada de deshabilitar la red

En este punto, hemos relacionado el hecho de tener binarios en nuestra lista con el hecho de haber detectado una extensión peligrosa (aparece en la *blacklist*).

Procedemos a comprobar cada uno de los binarios que tenemos en la lista si se encuentra en ejecución, ya que se tratará de un posible origen del fichero cifrado.

4.5 Detener de procesos sospechosos

Para esta tarea, usamos la función *KillProcessByName*, encargada de detener los procesos de la lista, que recibirá los nombres de los ejecutables detectados en el sistema.

```
Algoritmo KillProcessByName(Nombre_EXE)
    Repetir
        found = Falso
        Mientras Haya procesos Hacer
            proceso = OpenProcess()
            Si proceso.Nombre == Nombre_EXE Entonces
                found = Verdadero
                Si proceso.Propietario != "SYSTEM" Entonces
                    VolcadoMemoria(proceso)
                    FinalizarProceso(proceso)
                Fin Si
            Fin Si
        Fin Mientras
    Hasta Que found == Falso
FinAlgoritmo
```

Figura 35. Bucle sobre los procesos activos del sistema

En la Figura 35, podemos ver como iteramos sobre los procesos activos del sistema, en busca de un proceso que tenga como nombre el *Nombre_EXE*. En caso de encontrarlo, activamos el flag conforme lo hemos logrado e intentamos tomar el control del mismo. Para obtenerlo, tuvimos que activar el privilegio *SeDebugPrivilege*, reservado para tareas de debugging de funciones de sistema [20]. Este nos permite obtener el control de cualquier proceso en ejecución. En este punto, comprobamos el propietario del proceso mediante la función *GetProcessOwner* con el fin de evitar un falso positivo, ya que si el malware ha asignado un nombre protegido de sistema, podremos evitar detener un proceso crítico de sistema. Entendemos como proceso crítico de sistema los que se ejecutan con el usuario *SYSTEM*. Cabe mencionar que el malware (en circunstancias normales) siempre se ejecutará con el usuario que lo ha lanzado.

```
/* Comprobamos que el usuario no sea SYSTEM */
GetProcessOwner( hProcess,processOwner);
if (strcmp(processOwner, "SYSTEM") != 0 && pEntry.th32ProcessID != lastPID){
    WriteFullDump(hProcess,binaryName); //Realizamos el volcado de memoria del proceso
    TerminateProcess(hProcess, 9);
    sprintf_s(strLog,256,"Matando proceso %s PID: %d %s\n",
        pEntry.szExeFile,pEntry.th32ProcessID,processOwner);
    lastPID = pEntry.th32ProcessID;
    Log(strLog);
    res = 1;
}
CloseHandle(hProcess);
```

Figura 36. Detención de proceso sospechoso

Si el propietario no es *SYSTEM* procedemos a realizar un volcado de la memoria del proceso mediante la función *WriteFullDump*, con el fin de poder analizarlo posteriormente e intentar obtener información acerca del malware y, en el mejor de los casos, la clave simétrica de cifrado. Una vez realizamos el volcado, mandamos una señal *TerminateProcess* para detener el proceso y repetimos estos pasos sobre el resto de procesos activos del sistema por si hay más procesos con el mismo nombre. Se realizan barridos del árbol de procesos activos hasta que se encuentra totalmente libre de procesos sospechosos.

4.6 Rename del posible fichero malicioso

Una vez finalizado el barrido y detención de procesos los mismos, se procede a renombrar el fichero con la extensión *.virus* que usamos para inhabilitar una posible ejecución programada y/o no intencionada del mismo (la mayor parte del ransomware crea una tarea programada en sistema para ejecutar el malware de nuevo una vez reiniciado el equipo).

```
/* Renombramos el exe */
lastBinaryPath = lastBinaries[index,1];
if( lastBinaryPath->Contains(".")){
    char* binaryPath = (char*)Marshal::StringToHGlobalAnsi(lastBinaryPath).ToPointer();
    char* newBinaryPath2 = (char*)Marshal::StringToHGlobalAnsi(
        lastBinaryPath->Substring(0,lastBinaryPath->LastIndexOf("."))).ToPointer();
    const char * const SUFFIX = ".virus";
    size_t size = strlen(newBinaryPath2) + strlen(SUFFIX) + 1;  // +1 for NULL
    char* newBinaryPath = (char*)malloc(size);
    strcpy_s(newBinaryPath,size,newBinaryPath2);
    strcat_s(newBinaryPath,size,SUFFIX);
    if (rename(binaryPath, newBinaryPath) == 0){
        sprintf_s(strLog,256,"Binario malicioso renombrado %s\n",newBinaryPath);
        Log(strLog);
    }
    else{
        sprintf_s(strLog,256,"BINARIO NO Renombrado\n");
        Log(strLog);
    }
}
```

Figura 37. Renombramos el binario malicioso a .virus

En la Figura 37 vemos la función encargada de asignar el nuevo nombre al binario malicioso.

4.7 Shutdown del equipo

El último paso de nuestro proceso se encarga de lanzar una señal de *shutdown* al equipo afectado con el fin de informar al usuario de la detección realizada y evitar una mala gestión por su parte.

```
/* Lanzamos la señal de apagado del sistema */
system("shutdown -s -t 60 -c \"Se han detectado patrones de malware y se procede
    a apagar el equipo. POR FAVOR, NO LO ENCIENDA DE NUEVO.\" ");
sprintf_s(strLog,256,"Shutdown enviado\n");
Log(strLog);
index++;
break;
```

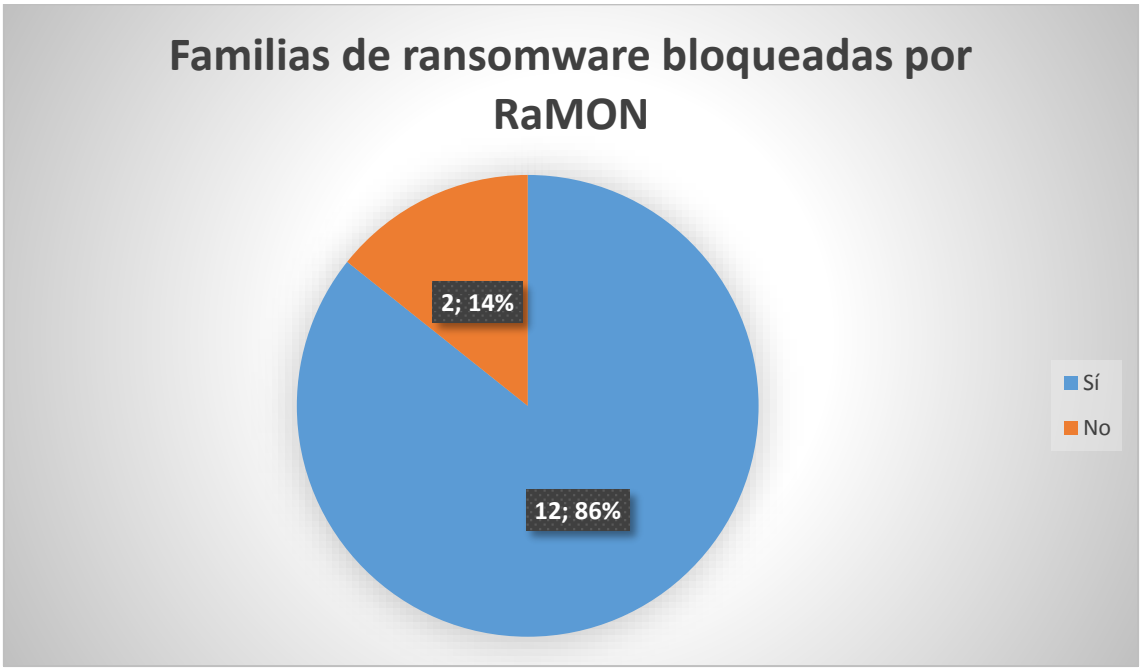
Figura 38. Señal de apagado y aviso al usuario infectado

Una vez realizado todo el proceso, el técnico dispondrá del fichero de logs donde se ha registrado todos los eventos realizados por RaMON y dispondrá además del volcado de memoria del proceso detenido. Esta información tiene como objetivo ayudar a reconstruir los hechos e identificar el posible vector de infección explotado.

5. Resultados

Al finalizar las pruebas con las diferentes muestras analizadas, los resultados fueron los siguientes:

14 familias analizadas



Las 2 familias de ransomware no detectadas por la aplicación presentaban las siguientes características:

- **Torrentlocker**: El proceso encargado de cifrar es un proceso legítimo de sistema al que se ha inyectado el código malicioso (explorer.exe). No se puede detectar el ejecutable.
- **Virlock:** Crea cientos de procesos para el cifrado y utiliza la extensión EXE para los ficheros cifrados, hecho que colapsa nuestro listado de ficheros ejecutables.

6. Problemas encontrados durante el desarrollo

Durante el desarrollo de la aplicación, hemos ido recogiendo los diferentes problemas que nos hemos ido encontrando en la siguiente tabla:

Problema	Solución
No siempre es el último proceso creado el que se encarga de cifrar.	Guardamos y procesamos los últimos 5 ejecutables creados en el sistema.
Se puede dar el caso de que el ransomware genere un ejecutable, pero no lo ejecute (en el momento del cifrado).	Antes de intentar finalizar un ejecutable, comprobamos si está en ejecución en el árbol de procesos del sistema.
A veces el proceso de cifrado está bloqueado por diversos mecanismos de protección (uso de nombres de proceso protegidos de sistema, creación de procesos dependientes que impiden la finalización del proceso padre,…)	Hemos implementado el método SeDebugPrivilege que otorga privilegios totales con el objetivo de poder finalizar cualquier proceso, saltándonos el bloqueo. *Explicado en el punto 4.5*
Hemos detectado ransomwares que utilizan nombres protegidos de procesos de sistema, hecho que nos provocaba la finalización de procesos legítimos y/o críticos.	Para corregir este problema, hemos añadido la comprobación del usuario que ejecuta cada proceso antes de proceder a su finalización. *Explicado en el punto 5.4*
(W7) Hemos detectado que si el binario se llama como un binario de sistema, RaMON lo registra como binario[1].exe y eso nos impide finalizar el proceso y renombrar el binario.	Hemos parcheado esta posibilidad. *Explicado en el punto 5.2.1*
En el proceso de deshabilitar las interfaces de red, nos encontramos con un "bug" en Windows XP que no permite esta acción en caso de solo existir una interfaz.	Para evitar este problema, hemos aplicado una configuración incorrecta de red, que inutiliza la interfaz y consigue bloquear el tráfico. *Explicado en el punto 5.3*
En Windows XP, el comando para gestionar dispositivos de red, requiere que introduzcamos el nombre de la interfaz para identificarla.	Hemos iterado por todos los dispositivos para capturar el nombre y aplicar los cambios. *Explicado en el punto 5.3*
La *blacklist* era estática y se tenía que actualizar y replicar manualmente a cada equipo para mantenerlos actualizados y protegidos	Hemos desarrollado un módulo de descarga del fichero de *blacklist*, de modo que lo tenemos actualizado, accesible, centralizado y gestionado desde el fichero de configuración de la aplicación. *Explicado en el punto 5.1.3*

Tabla 6. Problemas encontrados durante el desarrollo de RaMON

7. Posibles mejoras del proyecto

En este apartado vamos a enumerar las posibles mejoras que hemos detectado durante el a

- Pausar la ejecución del proceso supuestamente malicioso para que el usuario pueda confirmar su detención, en caso de proceder. En este punto se deberían resolver problemas referentes a las familias de ransomwares que crean múltiples procesos hijo para el cifrado. Esto supondría hacer que el proceso de monitorización fuera multihilo.
- Abarcar procesos hookeados de sistema
- Cuarentena del proceso para su posterior análisis en laboratorio.
- Aplicar la detección a ficheros que no sean solo de tipo .EXE
- Controlar el caso de un malware ejecutado como SYSTEM, que ha conseguido elevar privilegios mediante un EK.
- Añadir la opción "Generar dump del proceso" en el fichero de configuración. El mecanismo de generar el volcado de memoria del proceso malicioso antes de detenerlo causa siempre un mayor, aunque ligeramente, tiempo de finalización, y con ello, una mayor cantidad de ficheros cifrados. Si el entorno donde se instala la aplicación no tiene los recursos para analizar el volcado de memoria o simplemente no lo cree necesario, se debería poder desactivar.
- Añadir una whitelist de extensiones conocidas, con prioridad superior a la *blacklist*. De este modo, evitamos un posible falso positivo producido por la introducción incorrecta de una extensión maliciosa en la *blacklist*.
- Detección de ficheros ejecutables mediante la cabecera o el magic number, sustituyendo la actual detección por extensión.

8. Concordancia de resultados y objetivos

En su concepción inicial, se tomó como premisa que el uso de patrones como las extensiones fijas, patrones en las extensiones de los ficheros cifrados o las instrucciones para recuperar los mismos, nos podían servir para detectar, si no todos, una gran partes de los procesos de cifrado maliciosos.

Al finalizar el desarrollo del proyecto hemos podido verificar esta premisa aunque han aparecido nuevos aspectos que no habíamos tenido en cuenta inicialmente. Durante el periodo de tiempo que ha durado la investigación, han aparecido nuevas técnicas y otras que eran usadas por una minoría de familias de malware se han hecho cada vez más generalizadas.

El uso de librerías DLL y las inyecciones en procesos legítimos del sistema han ganado presencia en las nuevas variantes y familias de Ransomware, que con el paso del tiempo van mejorando los mecanismos de infección y evasión de la detección por parte de softwares como RaMON.

9. Conclusiones

En un inicio, la aplicación se pensó para aportar una solución alternativa al problema de la detección de un proceso de cifrado malicioso en un estado inicial, minimizando la pérdida de ficheros y el impacto del ataque.
Las soluciones que podíamos encontrar en el momento de iniciar el proyecto, o se basaban en premisas muy frágiles (como el cifrado por orden alfabético) o el tiempo de detección era tan elevado que las pérdidas eran cuantiosas (análisis heurístico y de entropía).

Planteamos una forma alternativa para la detección, basada en IOC's rápidas de detectar, como son las extensiones sospechosas y los ficheros donde se facilita la información para la recuperación de los archivos cifrados. La propuesta destacaba por la velocidad de detección, aunque relacionar esa detección con el proceso malicioso responsable del cifrado, era algo más difícil.

Durante el desarrollo se han descubierto nuevos comportamientos de este tipo de malware, haciendo necesaria la adaptación de RaMON durante el proceso. Se han añadido funcionalidades que no estuvieron contempladas en un inicio (como el volcado de memoria), ya que pueden aportar un valor añadido a la herramienta y creemos que es necesario aprovechar el mecanismo de detección para facilitar el análisis de este tipo de infecciones.
Una vez finalizado el desarrollo, hemos visto cumplidos los objetivos iniciales y parte de los descubiertos durante el proceso. Hemos confirmado la propuesta inicial, que mantenía que se podía lograr una contención significativamente más rápida, aun sacrificando un pequeño ratio de detección.

Nos ha servido para estudiar los comportamientos que ya conocíamos y, sobretodo, para aprender sobre los nuevos, sorprendiéndonos con la originalidad y técnica que, a su vez, son la base del continuo crecimiento en el mundo de la seguridad informática. Debido al continuo y creciente desarrollo en el mundo del malware, podemos dar cobertura a una parte relativamente pequeña de las amenazas. Esto supone la necesidad de evolucionar y trabajar duramente las herramientas para poder mantener un nivel adecuado de seguridad.

Cabe hacer especial hincapié en el hecho de que RaMON ha sido diseñado como una capa adicional y no debe sustituir nunca a las herramientas básicas, como son AntiVirus, Firewall, gestión adecuada de permisos de seguridad y políticas de grupo y, sobretodo, la educación y concienciación de los usuarios que, finalmente, son el eslabón más débil en la cadena de infección.

Índice de Figuras

Referencias

[1] Jaime Domenech. “El valor medio del rescate por ransomware sube hasta los 679 dólares” 20 Julio 2016.
<http://www.silicon.es/valor-medio-del-rescate-ransomware-sube-los-679-dolares-2314309>
[Consultado: 14 Diciembre 2016]

[2] Pierluigi Paganini. “Ransomware: A Highly-Profitable Evolving Threat” 8 Junio 2016.
<http://resources.infosecinstitute.com/ransomware-an-evolving-threat-even-more-profitable/>
[Consultado: 21 Junio 2016]

[3] Symantec. “Ransomware and Businesses 2016” 19 Julio 2016.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf>
[Consultado: 23 Agosto 2016]

[4] Kevin Savage, Peter Coogan, Hon Lau. “The evolution of ransomware” 6 de Agosto de 2015.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf> [Consultado: 3 Noviembre 2016].

[5] Cassius Puodzius. “How encryption molded crypto-ransomware” 17 Septiembre 2016.
<http://www.welivesecurity.com/2016/09/13/how-encryption-molded-crypto-ransomware/>
[Consultado: 12 Diciembre 2016]

[6] Symantec. “The Evolution of Ransomware” 6 Agosto 2015.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf> [Consultado: 25 Agosto 2016]

[7] Calyptix Blog. “Verizon Data Breach Report 2015” 11 Abril 2016.
<http://www.calyptix.com/research-2/verizon-data-breach-report-2015-top-10-charts-and-summary/>
[Consultado: 12 Mayo 2016]

[8] HelpSec. “Malware-infected home routers used to launch DDoS attacks” 12 Mayo 2015.
<http://www.helpsec.net/malware-infected-home-routers-used-to-launch-ddos-attacks>
[Consultado: 14 Diciembre 2016]

[9] Andra Zaharia. “What is Ransomware and 15 Easy Steps To Keep Your System Protected” 7 Julio 2016.
<https://heimdalsecurity.com/blog/what-is-ransomware-protection/>
[Consultado: 9 Agosto 2016]

[10] StatCounter. “Top 7 Desktop OSs from July 2015 to August 2016” 1 Septiembre 2016.
http://gs.statcounter.com/#desktop-os-ww-monthly-201507-201608>
[Consultado: 3 Septiembre 2016]

[11] StatCounter. “Top 7 Desktop OSs from 2016” 1 Septiembre 2016.
<http://gs.statcounter.com/#desktop-os-ww-monthly-201507-201608-bar>
[Consultado: 3 Septiembre 2016]

[12] Ed Bott. “Windows 10 usage share continues to grow, but enterprise stays on sidelines” 2 Febrero 2016.
<http://www.zdnet.com/article/windows-10-usage-share-continues-to-grow-but-enterprise-stays-on-sidelines/> [Consultado: 3 Septiembre 2016]

[13] MalwareBytes. “PRUEBA DE MALWAREBYTES 3.0 PREMIUM”
<https://es.malwarebytes.com/trial/#trial> [Consultado: 10 Diciembre 2016]

[14] Anand Khanse. “HitmanPro.Alert Review: Free Ransomware Protection” 8 Noviembre 2013. <http://www.thewindowsclub.com/hitman-pro-alert-review-free> [Consultado: 13 Noviembre 2016]

[15] Catalin Cimpanu. “Cryptostalker, a Tool to Detect Crypto-Ransomware on Linux” 22 Marzo 2016. <http://news.softpedia.com/news/cryptostalker-a-tool-to-detect-crypto-ransomware-on-linux-502002.shtml> [Consultado: 11 Diciembre 2016]

[16] Foolish IT. “CryptoPrevent Malware Prevention” 28 Mayo 2015. <https://www.foolishit.com/cryptoprevent-malware-prevention/> [Consultado: 20 Octubre 2016]

[17] Yago Jesús. “ANTIRANSOM V3” 21 Junio 2016. <http://www.securitybydefault.com/2016/06/anti-ransom-v3.html> [Consultado: 20 Octubre 2016]

[18] BitDefender. “Anti-Ransomware Tool” 29 Marzo 2016. <http://www.bitdefender.com/solutions/anti-ransomware-tool.html> [Consultado: 21 Octubre 2016]

[19] Amador Pérez Trujillo. “PROTEIN - PROTEcting your INformation” 27 Junio 2016. <https://gallery.technet.microsoft.com/scriptcenter/PROTEIN-PROTEcting-your-e4b11014> [Consultado: 21 Octubre 2016]

[20] Microsoft. “How to obtain a handle to any process with SeDebugPrivilege” 8 Junio 2009. <https://support.microsoft.com/es-es/help/131065/how-to-obtain-a-handle-to-any-process-with-sedebugprivilege> [Consultado: 27 Marzo 2016

Bibliografía

Trend Micro. “Por Qué Funciona el Ransomware” 28 Junio 2016.
<http://blog.la.trendmicro.com/por-que-funciona-el-ransomware-la-sicologia-y-los-metodos-utilizados-para-distribuir-infectar-y-extorsionar/#.WFCIpvnhDDc> [Consultado: 12 Agosto 2016]

Brooks Li. “Locky Ransomware Now Downloaded as Encrypted DLLs” 29 Agosto 2016.
<http://blog.trendmicro.com/trendlabs-security-intelligence/locky-ransomware-now-downloaded-encrypted-dlls/> [Consultado: 23 Septiembre 2016]

Andrea Allievi, Earl Carter and Emmanuel Tacheau. “TeslaCrypt – Decrypt It Yourself” 27 Abril 2015
<http://blogs.cisco.com/security/talos/teslacrypt> [Consultado: 15 Noviembre 2016]

Minhal Katri. “Ransomware Statistics – Growth of Ransomware in 2016” 25 Agosto 2016.
<http://blogs.systweak.com/2016/08/ransomware-statistics-growth-of-ransomware-in-2016/>
[Consultado: 14 Septiembre 2016]

J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten.
“Cold Boot Attacks on Disk Encryption” 21 Febrero 2008.
<http://citp.princeton.edu/research/memory/code/> [Consultado: 14 Noviembre 2016]

OSClass.org. “¿Cómo funcionan los hook?” 22 Junio 2012.
<http://forums.osclass.org/espanol-8/como-funcionan-los-hook/> [Consultado: 2 Octubre 2016]

Javier Olmedo. “Herramienta para recuperar archivos cifrados por ransomware TeslaCrypt” 2 Mayo 2016.
<http://hackpuntes.com/herramienta-para-recuperar-los-archivos-cifrados-por-el-ransomware-teslacrypt/> [Consultado: 27 Agosto 2016]

Pierluigi Paganini. “New Locky Ransomware variant uses DLLs for distribution” 27 Agosto 2016.
<http://securityaffairs.co/wordpress/50669/malware/new-locky-ransomware-dlls.html>
[Consultado: 28 Agosto 2016]

Vencislav Krustev. “Use Wireshark to Decrypt Encoded Files by Ransomware” 19 Mayo 2016.
<http://sensorstechforum.com/use-wireshark-decrypt-ransomware-files/>
[Consultado: 2 Septiembre 2016]

Lawrence Abrams. “Petya and Mischa Ransomware Affiliate System Publicly Released” 26 Julio 2016.
<http://www.bleepingcomputer.com/news/security/petya-and-mischa-ransomware-affiliate-system-publicly-released/> [Consultado: 12 Septiempre 2016]

Rosalía Arroyo. “Exploit Kits, o conviértete en un hacker” 21 Julio 2015.
<http://www.channelbiz.es/2015/07/21/exploit-kits-o-conviertete-en-un-hacker>
[Consultado: 27 Diciembre 2016]

Christiaan Beek and Andrew Furtak. “Targeted Ransomware No Longer a Future Threat” Febrero 2016.
<http://www.intelsecurity.com/advanced-threat-research/content/Analysis_SamSa_Ransomware.pdf>
[Consultado: 26 Agosto 2016]

McAfee Labs. “Threads Report” Septiembre 2016.
<http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-sep-2016.pdf>
[Consultado: 2 Octubre 2016]

McAfee Labs. “2016 Threats Predictions” Octubre 2015.
<http://www.mcafee.com/us/resources/reports/rp-threats-predictions-2016.pdf>
[Consultado: 2 Octubre 2016]

Pablo San Emeterio. “WHF WINDOWS HOOKING FRAMEWORK” 23 Abril 2013.
<http://www.securitybydefault.com/2013/04/whf-windows-hooking-framework-i.html>
<http://www.securitybydefault.com/2013/04/whf-windows-hooking-framework-ii.html>
[Consultado: 4 Septiembre 2016]

Symantec. “Ransomware and Business 2016” 6 Mayo 2016.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf> [Consultado: 26 Agosto 2016]

Gavin O’Gorman and Geoff McDonald. “Ransomware: A Growing Menace” 8 Noviembre 2012.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ransomware-a-growing-menace.pdf> [Consultado: 26 Agosto 2016]

Kevin Savage, Peter Coogan and Hon Lau. “The evolution of ransomware” 6 Agosto 2015.
<http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf> [Consultado: 26 Agosto 2016]

Hasherezade. “Look Into Locky Ransomware” 1 Marzo 2016.
<https://blog.malwarebytes.com/threat-analysis/2016/03/look-into-locky/>
[Consultado: 7 Agosto 2016]

Edgar Vásquez Cruz. “Ransomware, una amenaza renacida y negocio lucrativo” 6 Junio 2016.
<https://blogs.mcafee.com/languages/espanol/ransomware-una-amenaza-renacida-y-negocio-lucrativo/> [Consultado: 8 Agosto 2016]

Microsoft Malware Protection Center. “The 5Ws and 1H of Ransomware” 18 Mayo 2016.
<https://blogs.technet.microsoft.com/mmpc/2016/05/18/the-5ws-and-1h-of-ransomware/>
[Consultado: 8 Agosto 2016]

Alexs Pena. “How to Deal with Ransomware” 6 Abril 2016.
<https://blogs.technet.microsoft.com/office365security/how-to-deal-with-ransomware/>
[Consultado: 8 Agosto 2016]

Andra Zaharia. “What is Ransomware and 15 Easy Steps To Keep Your System Protected” 7 Julio 2016.
<https://heimdalsecurity.com/blog/what-is-ransomware-protection/> [Consultado: 9 Agosto 2016]

Fedor Sinitsyn and Victor Alyushin. “The Shade Encryptor: a Double Threat” 14 Septiembre 2015.
<https://securelist.com/analysis/publications/72087/the-shade-encryptor-a-double-threat/>
[Consultado: 20 Octubre 2016]

Lawrence Abrams. “TeslaCrypt shuts down and Releases Master Decryption Key” 19 Mayo 2016.
<https://www.bleepingcomputer.com/news/security/teslacrypt-shuts-down-and-releases-master-decryption-key/> [Consultado: 16 Octubre 2016]

Vadim Kotov and Mantej Singh Rajpal. “Understanding Crypto-Ransomware” 14 Noviembre 2014.
<https://www.bromium.com/sites/default/files/rpt-bromium-crypto-ransomware-us-en.pdf>
[Consultado: 20 Agosto 2016]

Zeljka Zorz. “TeslaCrypt: New versions and delivery methods, no decryption tool” 22 Abril 2016.
<https://www.helpnetsecurity.com/2016/04/22/teslacrypt-new-versions-no-decryption/>
[Consultado: 3 Septiembre 2016]

Osterman Research. “Understanding the depth of the global ransomware problem” Agosto 2016.
<https://www.malwarebytes.com/surveys/ransomware/?aliId=13242065>
[Consultado: 7 Septiembre 2016]

Bryan Lee. “Ransomware: Unlocking the lucrative criminal business model” 10 Mayo 2016. <https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/research/ransomware-report> [Consultado: 12 Septiembre 2016]

James Wyke and Anand Ajjan. “Current state of Ransomware” 9 Diciembre 2015. <https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-current-state-of-ransomware.pdf> [Consultado: 17 Agosto 2016]

Anexo 1: Tabla comparativa herramientas Anti-Ransomware vs RaMON

	RaMON	Anti Ransom	Protein	Hitman Pro	BitDefender AntiRansomware	MalwareBytes Anti-Ransomware	CryptoPrevent	AppLocker
Tecnología Detección	Extensiones fijas	Canary Files	Monitorización de llamadas a funciones sospechosas	Monitorización de llamadas a funciones sospechosas	Monitorización de llamadas a funciones sospechosas	Monitor heurístico de comportamiento	GPO	GPO
Compatible WindowsXP	✓	✗	✗	✓	✓	✓	✓	✗
Gratuito	✓	✓	✓	30 días	✓	✗	✓	✓
Alcance	Todo	Todo	Todo	Todo	Locky, TeslaCrypt y CTB-Locker	Todo	CryptoLocker	Todo
No requiere soft. adicional	✓	✗	✗	✓	✗	✗	✓	✓
Capa Independiente	✓	✓	✓	✓	✓	✗	✓	✓
Peso	37KB	38.8MB	72KB	11MB	4.5MB	36MB	2.5MB	0MB
No requiere instalación	✓	✗	✓	✓	✗	✗	✓	✓

Anexo 2: Diagrama de flujo